National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Decentralized Control of Multiple Robots Moving in Formation

BY

Wenzhong Tang ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Masters of Science.

DEPARTMENT OF COMPUTING SCIENCE

Edmonton, Alberta
Fall 1995

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

THE AUTHOR HAS GRANTED AN IRREVOCABLE NON-EXCLUSIVE LICENCE ALLOWING THE NATIONAL LIBRARY OF CANADA TO REPRODUCE, LOAN, DISTRIBUTE OR SELL COPIES OF HIS/HER THESIS BY ANY MEANS AND IN ANY FORM OR FORMAT, MAKING THIS THESIS AVAILABLE TO INTERESTED PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE IRREVOCABLE ET NON EXCLUSIVE PERMETTANT A LA BIBLIOTHEQUE NATIONALE DU CANADA DE REPRODUIRE, PRETER, DISTRIBUER OU VENDRE DES COPIES DE SA THESE DE QUELQUE MANIERE ET SOUS QUELQUE FORME QUE CE SOIT POUR METTRE DES EXEMPLAIRES DE CETTE THESE A LA DISPOSITION DES PERSONNE INTERESSEES.

THE AUTHOR RETAINS OWNERSHIP OF THE COPYRIGHT IN HIS/HER THESIS. NEITHER THE THESIS NOR SUBSTANTIAL EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT HIS/HER PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE DU DROIT D'AUTEUR QUI PROTEGE SA THESE. NI LA THESE NI DES EXTRAITS SUBSTANTIELS DE CELLE-CI NE DOIVENT ETRE IMPRIMES OU AUTREMENT REPRODUITS SANS SON AUTORISATION.

ISBN 0-612-06545-6

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Wenzhong Tang

TITLE OF THESIS: Decentralized Control of Multiple Robots Moving in Formation

DEGREE: Masters of Science

YEAR THIS DEGREE GRANTED: 1995

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) Wenzhong Tang

Wenzhong Tang
#2A, 9108 – 112 Street
Edmonton, Alberta
Canada, T6G 2C5

Date: June 5, 1995

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Decentralized Control of Multiple Robots Moving in Formation** submitted by Wenzhong Tang in partial fulfillment of the requirements for the degree of Masters of Science.

Hong Zhang

Dr. Hong Zhang (Supervisor)

Dr. Max Q.-H. Meng (External)

Dr. Anup Basu (Examiner)

Dr. Keith Smillie (Chair)

Keith Smillie

Date: 5/30/95

To my parents

Abstract

In achieving collective robot tasks, it is often required that a group of autonomous mobile robots maintain a certain formation while moving along a specified path towards a goal position. This is referred to as the *formation marching* of a multi-robot group.

Several approaches have been proposed for the formation marching problem. However, these approaches rely on centralized control or extensive global communications among robots; as a result, they are unsuitable for large robot groups or robot groups working in hostile or extreme environments. In this thesis, a decentralized control approach to the formation marching problem using the *neighbor following* method is developed from studying the formation flight of large birds. The feasibility of the neighbor following have been demonstrated through simulation.

Acknowledgements

First, I would like to thank my supervisor, Dr. Hong Zhang, for his guidance throughout this research. This thesis would not come into existence without his numerous suggestions. I would also like to thank Dr. Anup Basu and Dr. Max Q.-H. Meng for reviewing my thesis and serving as my examine committee members and Dr. Keith Smillie for serving as the chairman of my Master's defense.

Special thanks to Ron Kube and Yaoqing Gao for reviewing drafts of this thesis and providing many valuable comments. Thanks also to Ning Chen, Michael Boshra, Jun Tang, and Xianchang Wang who provided valuable suggestions on my research.

Finally, I would like to thank Yan Wei, not only for her verifying mathematical deductions, proofreading numerous drafts, and helping in using MATLAB, but also for her love and encouragement. Thanks for the confidence in me!

Contents

1 Introduction 1

1.1 Motivation of Work 1

1.2 Structure of the Thesis 4

2 Related Topics 6

2.1 Introduction 6

2.2 Behavior-based Robotics 7

2.2.1 Subsumption Architecture 8

2.2.2 Motor Schema Based Reactive Control 9

2.3 Collective Robotics 10

2.3.1 Collective Behavior and Swarm Intelligence 11

2.3.2 Classifications of Collective Tasks and Multi-robot Systems . . 13

2.4 Decentralized Control Architectures for Collective Robots 14

2.5 Inter-robot Communications 15

2.6 Self-Organization and Formation Generation 17

2.7 Previous Research in Decentralized Formation Marching 19

2.7.1 Biological Evidence 19

2.7.2 Previous Research 22

2.8 Summary 25

3 The Neighbor Following Approach **26**

3.1 Introduction 26

3.2 The Neighbor Following Approach to the Formation Marching of Multiple Robots 27

3.2.1 Definitions 27

3.2.2 What is Neighbor Following? 28

3.2.3 The Robot Model 30

3.2.4 Single Neighbor Following Algorithm 33

3.2.5 Eliminate Static Formation Error by Dynamic Prediction . . . 35

3.2.6 Leader Direction and Speed Estimation Using Position Measurements 39

3.3 Multiple Neighbor Following 41

3.4 Neighbor Following Sequence 42

3.5 Summary 43

4 Implementation **45**

4.1 Introduction 45

4.2 The Architecture of RoboSquad 46

4.3 Robot Modeling and Implementation 47

4.3.1 Sensors 48

4.3.2 Behaviors 49

4.3.3 Actuators 53

4.4 User Interface 53

4.5 Summary 57

5 Simulation Results **58**

5.1 Introduction 58

5.2 Formation Error 60

5.3 Error Sources in Physical Implement 62

5.3.1 Robot Sensors 62

5.3.2 Sensor Errors 64

5.4 The Effects of Sensor Errors on the Performance of Robot Formation 65

5.5 The Effects of Local Communication on the Performance of Robot Formation 66

5.6 Single Neighbor Following vs. Multi-Neighbor Following 71

5.6.1 Formation Performances under Noisy Sensor Inputs 71

5.6.2 Formation Performances in Obstacle Filled Environment . . . 73

5.7 The Effects of Different Following Sequences on the Performance of Robot Formation 74

5.8 Summary 75

6 Conclusion **79**

6.1 Summary of Thesis 79

6.2 Future Works 80

Bibliography **82**

List of Figures

1 Line formations. A: column; B: front; C: echelon. 20

2 Line formations. A: J; B: V. 20

3 Line formations. A: inverted J; B: inverted V. 21

4 Line and compound formations. A: closed line; B: branched V. 21

5 A simple formation of 6 robots. 28

6 Physical structure of a mobile robot. 30

7 Spatial Information needed for a robot R_f to keep a specific position with respect to another robot R_l. 31

8 Calculate follow behavior of a follower robot 33

9 Simulation snapshots of 3 robots moving in a column formation without using dynamic prediction. Upper left: step 0; Upper right: step 172; Lower left: step 351; Lower right: step 598. 37

10 Simulation snapshots of 3 robots moving in a column formation using dynamic prediction. Upper left: step 0; Upper right: step 172; Lower left: step 351; Lower right: step 598. 38

11 Direction and speed estimation using two position measurements . . . 39

12 A follower robot R_f decides its motion based on the motions of its two leaders R_{l1} and R_{l2}. F_1 and F_2 are the formation forces produced by formation keeping algorithms associated with F_1 and F_2, respectively, and F_f is the resultant formation force. 41

13 Different Neighbor following architectures. Each numbered circle in the figure represents a robot with a different ID, and each arrow represents a follow relationship from a robot to another robot. A: single neighbor following; B: multiple neighbor following 43

14 Structure of RoboSquad 46

15 Structure of the robot part of RoboSquad 47

16 Graphical user interface of RoboSquad 54

17 Simulation snapshots showing a group of 3 robots make a right turn in an obstacle-free environment 55

18 Simulation snapshots showing a group of 3 robots make a right turn through various obstacles 56

19 Three different robot formats. A: Line formation with 3 robots; B: Grid formation with 6 robots; C: Pyramid formation with 10 robots. . 59

20 Three different robot group moving trajectories. A: Straight line ("1") trajectory; B: "L" trajectory; C: "8" trajectory. 60

21 Sensor errors in measuring the relative distance and angle of a leader robot. 64

22 Three different robot groups and their neighbor following sequences. A: Three robots in a line formation; B: Six robots in a grid formation; C: Ten robots in a pyramid formation. 66

23 Average formation errors in three different robot group moving in three different trajectories. Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation. 67

24 Average formation errors in three different robot group moving in three different trajectories (with direction and speed estimation). Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation. 68

25 Average formation errors in three different robot group moving in three different trajectories. Communication is used but error in leader directions exist. Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation. 70

26 Different following sequences of a group of 6 robots with 1 group leader 71

27 Average formation errors in 3 different neighbor following sequences: top left: single neighbor following; top right: multiple neighbor following; bottom: complete neighbor following. 72

28 Two neighbor following sequences of a group of 6 robots. Left: single neighbor following; Right: multiple neighbor following. 74

29 Simulation snapshots of a group of 6 robots moving through obstacles using two different multiple neighbor following sequences: Top: single neighbor following; Bottom: multiple (two) neighbor following. 77

30 Different single neighbor following sequences of a group of 6 robots with 1 group leader 78

31 Average formation errors in 4 different single neighbor following sequences. 78

Chapter 1

Introduction

1.1 Motivation of Work

In recent years, extensive research has been carried out in autonomous mobile robots. Traditionally, research in autonomous mobile robots has concentrated on accomplishing tasks with a *single* robot. However, the creation of highly intelligent robots that are able to interact with the real world is still out of reach, and the task achieving abilities of any single robot are limited. Currently, there is an increasing interest in multi-robot systems. By using cooperative robot groups, it becomes possible to solve large scale problems that require more resource than any single robot can possibly provide. For tasks that can be achieved by one single robot, it is often faster, more reliable and more cost-effective to use cooperative robot groups.

Collective behaviors in social insects, bird, fish, and many land animals have been investigated by researchers in biology and behavior science for decades. Motivated by the group behaviors in animal societies and advances in related areas such as computer science, robotics, and complex systems, *collective robotics* (or *cooperative robotics*) tries to develop decentralized control techniques that enable multiple behavior-based autonomous robots to achieve tasks cooperatively.

In achieving collective tasks using a society of robots, the robots are usually required to remain in a group. This is achieved by the *group* behavior of each robot in the robot society. Normally, the group behavior only tries to keep each robot within some distance with respect to other robots, and there is no requirements on the exact geometric relationship between robots. If the robots in a multi-robot group are required to form and maintain a rigid geometric pattern while performing a collective task, the control of the robot group is much more complex. Nonetheless, it is useful in some multi-robot applications such as material transportation and is an interesting problem to study.

Two related but different research issues that have been considered are the *formation generation* problem and the *formation marching* problem. Formation generation or pattern formation is concerned with producing a desired spatial formation from a possibly arbitrary initial configuration. The problem of formation generation has been investigated using various methods including cellular automata [Wang1989], dynamic system analysis [Sugihara1990], potential field [Unsal1993, Genovese1992], etc. On the other hand, the formation marching problem, with which this thesis is concerned, assumes that the formation of the robots has already been created and the objective is for the formation to move through a given trajectory while maintaining the desired spatial configuration.

Formation marching is a common group behavior in many kinds of animals, especially in large birds, for example, Canada Geese. When migrating, these birds usually choose to fly in formations such as line or V formations. Researchers have argued that formation flight may have advantages in aerodynamics or effective use of sensory systems, but there is still no clear explanation on the mechanism of the formation flight.

Formation marching of multiple vehicles is often seen in military operations; for example, fleets of battleships or groups of aircrafts often maintain some fixed formation to use their sensing, attacking and defensing abilities efficiently. Also, in material

handling, sometimes multiple vehicles are required to distribute themselves equally around or under a load that any single vehicle is unable to handle. Similar potential applications can also be found in the areas of space and deep-sea exploration.

Formation marching of multiple autonomous mobile robots has similar applications with manned vehicles. In addition, robot groups can be used in hostile or extreme environments that are unsuitable for manned vehicles. However, appropriate methods must be designed to coordinate the motions of the robots.

Various approaches have been proposed to control robots moving in formation [Wang1991, Parker1993, Chen1994a, Chen1994b, Balch1994a, Brock1992]. These approaches use centralized or semi-centralized control and rely on inter-robot communication to achieve the objective. However, centralized control and extensive communication are not suitable for controlling large robot groups because of their high expenses; they are also not suitable for robot groups working under hostile or extreme conditions where communication is not possible or is not allowed. To overcome these problems, a decentralized control strategy is needed for formation marching of multiple robots.

In this thesis, a decentralized control approach – the *neighbor following* approach is introduced. The neighbor following approach has several distinct advantages compared with various other approaches:

- Satisfactory formation performance can be achieved without centralized control or extensive world knowledge;
- Inter-robot communication is limited to prevent communication bottleneck. Also communication between robots can be eliminated provided sensory information is accurate enough, thus making the neighbor following approach suitable for controlling large robot groups;

Simulation experiments have shown that robots groups using the neighbor following strategy can successfully maintain arbitrary formations while moving along a specified

path and avoiding obstacles. Also, neighbor following can be used in large robot groups and can be implemented in real robot systems easily.

1.2 Structure of the Thesis

The rest of the thesis contains 5 chapters. Chapter 2 discusses topics related to the decentralized control of multiple robots moving in formation. First, the behavior-based approach to build mobile robots is introduced, with emphasis on the subsumption architecture and reactive control. Second, concepts including collective behavior, collective robotics and swarm intelligence are introduced. Then, research in decentralized control of multiple cooperative robots and inter-robot communication is discussed. After introducing self-organization and pattern formation of multiple robots, previous research for formation marching in both biology and robotics domains are presented.

Chapter 3 introduces our decentralized control approach for the formation marching problem. In this chapter, the neighbor following approach including single neighbor following algorithm and multiple neighbor following algorithm is introduced. Error analysis is conducted to examine the relationship between sensing errors and formation performance and the effect of local communications. and the performance improvements using multiple neighbor following and dynamic prediction are introduced.

Chapter 4 introduces the robot population simulator – RoboSquad designed for research in formation marching of multiple robots. First, the design objective of RoboSquad is defined. Second, the modeling of the physical properties, sensory properties, and motion properties of robots are discussed. Then the implementations of individual behaviors and their integration are presented. Also, some features of the user interface of RoboSquad are discussed.

Chapter 5 compares the formation performances of robot groups under different

neighbor following algorithms and different simulation environments. How the existence of sensory errors, use of local communications, existence of obstacles, and different following sequences affect the performance of different algorithms is examined. Also, how multiple neighbor following can improve formation performance is demonstrated.

Finally, Chapter 6 summarizes our research results and outlines some future research directions.

Chapter 2

Related Topics

2.1 Introduction

Motion coordination of multiple robots moving in formation involves the control of many autonomous mobile robots. The computation complexity for centralized hierarchy control will increase exponentially with the increase of the number of robots. Collective robotics, as a decentralized control approach for multiple robots, attempts to achieve tasks collectively by invoking appropriate collective behaviors on the group of behavior-based robots. In conclusion, to study the formation marching of multiple robots, many issues such as behavior-based robotics, collective behavior, decentralized control and self-organization behaviors of multiple robots need to be addressed. Also, previous research in the formation marching problem both in biology domain and in robotics domain need to be investigated.

In this chapter, topics related to the research in the formation marching of multiple robots are discussed. In Section 2.2, concepts of behavior-based robotics are introduced. Emphasis will be put on Brooks' subsumption architecture and Arkin's reactive control. In Section 2.3, the notions of collective behavior and swarm intelligence are introduced, and classifications of collective tasks and multi-robot groups are presented. Section 2.4 discusses previous research in decentralized control of mul-

tiple cooperative robots Section 2.5 discusses some advantages and disadvantages of inter-robot communications. Section 2.6 introduces the self-organization and pattern formation of multiple robots. Finally, in Section 2.7, biological evidence on the formation flight of birds and previous proposed approaches on the formation marching of multiple robots are discussed.

2.2 Behavior-based Robotics

The final goal of Artificial Intelligence is the creation of artificial entities that display human level intelligence. Robotics, originally developed as a subfield of Artificial Intelligence, shares the common goal. In the past several decades, the field of robotics has made significant progresses. especially in the theories and applications in industrial automation. However, the attempt of building intelligent robots that can successfully interact with the dynamic world has failed to generate satisfactory results.

In the mid 80's, many researchers became interested in alternative approaches in producing intelligence and intelligent robots. They thought that an intelligent robot should be able to react to the dynamic aspects of the world and generate behaviors robust to inaccurate sensory data and unpredicted changes in the environment. They suspected that the *symbol system hypothesis* upon which classical AI is based is fundamentally flawed, and as such imposes severe limitations on its progress [Brooks1990]. Inspired by the advances in computer science, neuroscience, and biology, they proposed the concept of behavior-based robotics.

In contrast to traditional *SMPA* (sense–model–plan–act) based robots, a behavior-based mobile robot has a much tighter connection between sensing and actuation. A behavior-based mobile robot determines its activity by invoking some basic *emergent behaviors* like avoid, wander, and explore. Each behavior is a reaction of the robot to the dynamic world based on simple data inputs.

Many approaches to control behavior-based mobile robots have been proposed [Brooks1986, Arkin1987, Payton1990, Beer1990, Gat1992]. Among them Brooks' subsumption architecture and Arkin's motor schema based reactive control approach are widely adopted by researchers. A brief review of these two approaches is given in the following subsections.

2.2.1 Subsumption Architecture

The *subsumption architecture* for controlling autonomous mobile robots was first introduced in [Brooks1986] and further discussed in [Brooks1990] and [Brooks1991a]. It is probably the most well known behavior-based mobile robot architecture. In the subsumption architecture, the controller is built as a series of incremental layers, each layer is composed of a fixed-topology network of simple finite state machines augmented with timing elements, and this network of asynchronous *augmented finite state machines*(AFSMs) produces a simple task achieving behavior. New layers are added on top of existing layers. Layers communicate over low-bandwidth channels, and higher layers can "suppress" or "inhibit" behaviors produced by lower layers (subsumption). Since subsumption architecture is based on decomposition of a mobile robot in terms of behaviors rather than functional modules, the controller can be viewed as a system of separately acting agents, so there is no need for a central control module or internal representation of the world.

[Connell1990] describes a robot successfully built with the subsumption architecture and shows how seemingly goal-directed behaviors emerge from the interactions of simpler non goal-directed behaviors. The robot, Herbert, operates in an unmodified office environment occupied by moving people. Herbert's controller is composed of over 40 separate processes that run on a loosely connected network of 24 processors. Herbert can wander around office areas, go into people's offices and "steal" empty soda cans from their desks. Connell also provides several enhancements to Brooks' original design in the way the control system is layered. Brooks' layers define a total

order on the behaviors of a robot, whereas Connell's approach only defines a tree-like partial order. Also, Connell's approach does not require the priorities of various layers to follow their evolutionary sequence; even various parts of a layer may have different priorities relative to other existing layers they interact with.

The subsumption architecture can achieve robust performance for mobile robots in dynamically changing environments, and its hardware implementation is simple. However, it has the problem of lacking modularity and is difficult to form long-term goals because of its pure reactive nature. These problems lead researchers to seek hybrid control architectures that incorporate both deliberative and reactive behaviors.

2.2.2 Motor Schema Based Reactive Control

Behavior-based control or *reactive control* based on task decomposition is characterized by a stimulus-response type of relationship between a mobile robot and the world. It has been proved to be an effective means for producing robust performance in complex domains. However, purely reactive systems are incapable of formulating and following long-term goals because they are always immediately reacting to the world. [Arkin1990] argues that by incorporating various forms of knowledge using explicit internal representations, reactive control can be made considerably more flexible.

In [Arkin1987, Arkin1990], Arkin introduced the *motor schema based reactive navigation* approach. A *motor schema* corresponds to a primitive behavior that reacts to sensory information obtained from the environment, and more complex behaviors can be obtained by combining multiple simple motor schemas. Some primitive motor schemas are introduced including *Move-ahead*, *Move-to-goal*, *Avoid-static-obstacle*, *Stay-on-path*, *Dock*, *Noise*, *Move-up*, *Move-down*, and *Maintain-altitude*. Each primitive motor schema is instantiated as a separate asynchronous computing agent with parameters reflecting current world knowledge. The output of each individual motor schema is a velocity vector representing the direction and speed at which the robot is

to move given current environmental conditions. Potential field method[Khatib1986] is used to compute the output of each motor schema. It is worth noticing that the entire potential field is never computed by the robot, and only the point where the robot is currently located is computed, so the cost for computing each schema is small. The output of each primitive motor schema is combined using vector summation and normalization (keeping the resultant vector within the constraints of the robot's capabilities). To overcome possible local minima produced by potential fields, a *Noise* schema is used to produce a small vector that has a random direction. Also, gains (weights) of schema outputs can be changed depending on established real-time deadlines for goal attainment to allow a blocked robot to bypass obstacles.

In contrast with Brooks' statement that *the world is it's own best model* [Brooks1991a, Brooks1991b], Arkin argues that world knowledge plays an important role in a robot's interaction with the world. Although it is not a prerequisite for reactive control, it is a prerequisite for *efficient* and *flexible*, intelligent control techniques. Two types of world knowledge can be used. *Persistent knowledge* is the *a priori* information about the robot's environment that can be considered static for the duration of the mission, and *transitory knowledge* is dynamically acquired by the robot as it moves in the world. These two kinds of world knowledge, when combined with reactive control, will increase the capability of the robot interacting with the dynamic world.

The feasibility and flexibility of the motor schema based control has been proved by both simulation and real robot experiments. Also, it has been extended into multi-robot domain [Arkin1992, Balch1994b], which will be discussed in the next section.

2.3 Collective Robotics

Collective robotics is the research of collective task-achieving behaviors of multiple robots. Recent interests in collective robotics are mainly motivated by the following reasons:

- The creation of highly intelligent robots that are able to interact with the real world is still out of reach, and the tasks that can be achieved by a single robot are limited because of the primitive intelligence of current autonomous robots;
- Cooperation between multiple robots is essential for the successful completion of many tasks because these tasks require more resources than a single robot can possibly provide. By enabling multiple robots to cooperate on a large task, large-scale tasks that are often infeasible using a single robot can be solved;
- There is an apparent parallel between multiple robots and multiple insects/animals working as a group when the animat approach of constructing intelligence is used. Findings in the collective behaviors in the animal world can easily lead to control strategies in the robot world.

In this section, the concept of collective behavior is first introduced, followed by the concepts of collective robotics and swarm intelligence. Finally, taxonomies of eligible collective tasks and multiple robot systems are described.

2.3.1 Collective Behavior and Swarm Intelligence

A *collective behavior*, or *group behavior*, is a task-achieving activity of a group of agents that emerges from the activities of its individuals, and it consists of a common set of rules for accomplishing the task.

Social insects such as ants, bees, termites, and many kinds of birds, fish, and land animals display amazing collective task achieving abilities. As [Franks1989] pointed out,

> "In colonies of army ants, perhaps to a greater extent than in any other animal except man, we see the emergence of flexible problem solving far exceeding the capacity of the individual."

Through mass sensing, communication, and mass action, social insects and social animals are able to display a wide range of collective behaviors that are critical in the survival of their societies. [Kube1992b] pointed out that collective behaviors, just appear to be some random and often aberrant activities of a non-intelligent agent reacting to external environment when viewed at individual level. However, when viewed at the society level, they become an emergent property of a self-organizing system with a few simple interaction rules between individuals and between an individual and the external world. For social insects, mass communication is often used to incite common behaviors of individuals that result in a collective behavior of the group as the information propagates throughout the society. Many collective behaviors have been investigated by researchers. [Franks1989], for example, investigated the collective raid behavior of army ants. The formation moving behavior of birds that serves as one motivation of the study of formation marching of multiple robots is also investigated by many researchers.

Since the creation of highly autonomous robots that are able to work in real world is still out of reach, researchers have proposed methods to organize many simpler robots into task-achieving groups [Dario1991, Arkin1992, Kube1992a]. The use of multi-robot groups enlarges the robot applications to large-scale tasks that can not be achieved by individual robots, and improves the reliability of the task-achieving precess. However, there is one question: how do we develop task-achieving collective behaviors for a group of simple robots?

Based on the observations from social insects, [Kube1992b] claimed that control of robot populations can be achieved by invoking collective behaviors using different mechanisms such as common goal oriented collective task, environmental cues, and the awareness of other robots in the group. By invoking collective behaviors, groups of non-intelligent, behavior-based robots may exhibit collectively intelligent behaviors, thus *swarm intelligence* [Beni1992], or *collective robotic intelligence* [Kube1992b] can be achieved.

2.3.2 Classifications of Collective Tasks and Multi-robot Systems

Using groups of mobile robots to achieve collective tasks is the objective of collective robotics. [Kube1994] divided possible multi-robot applications into two categories, namely, *noncooperative tasks* and *cooperative tasks*. Noncooperative tasks can be achieved by an individual robot but using multiples can accomplish the task faster and more reliable. Examples of noncooperative tasks include searching [Genovese1992], foraging [Arkin1992], and cleaning. Cooperative tasks, on the other hand, can only be achieved by many robots working cooperatively. Examples of cooperative tasks include construction [Ueyama1992], transporting [Stilwell1994, Kube1994], pattern generation[Sugihara1990, Wang1989], and formation marching [Parker1993, Wang1991, Chen1994a], which will be further discussed later in the thesis.

[Dudek1993] made a taxonomy for swarm robots (multi-robot groups) to clarify the strengths, constraints, and tradeoffs of various multi-robot system designs, and to highlight various alternatives in designing such systems. The swarms are classified by:

- Swarm size, i.e., the number of robots in the environment (one robot, a pair of robots, small group, and large group).
- Communication range (no communication, small-range communication, global communication).
- Communication topology (broadcast, communication by address or name, tree hierarchy, graph-like structure);
- Communication bandwidth (free communication, low-cost, high-cost, no communication).
- Swarm reconfigurability, i.e., how easy a robot swarm can spatially re-organize itself (static arrangement, coordinated rearrangement, dynamic arrangement).

- Swarm unit processing ability (non-linear summation unit, finite state machines, push-down automation, Turing machine equivalent).
- Swarm composition (homogeneous, heterogeneous) .

This permits comparative analysis of different multi-robot systems and provides a framework for the analysis of the advantages and abilities of different collective robotic architectures. Resently, researchers have developed tools for the formal specifications of collective robots societies. For a good overview please see [MacKenzie1993]

2.4 Decentralized Control Architectures for Collective Robots

One important issue in collective robotics is the control strategy for coordinating multiple robots. Many multi-robot systems use master-slave hierarchies to coordinate the activities of robots [Noreils1990, Fukuda1989]. Although the master-slave hierarchy often allows the robot group to conduct a particular task more efficiently, the robustness of the group suffered. [Arkin1992] pointed out that there are three major drawbacks in the above strategy:

- Communication bottleneck exists when a master is trying to coordinate the behavior of a swarm of slave agents;
- Robustness adversely affected because the failure of one (master) robot may cause the failure of the entire system;
- Tradeoff between reliability in achieving collective tasks and efficient use of sensing, communication, and computation resource.

In [Arkin1992], The schema-based navigation is extended to multiple homogeneous robots working without central coordination and inter-robot communication. Its ef-

ficiency in retrieving objects in an unstructured environment is proved through simulation.

[Kube1994] Also investigated decentralized control methods for multiple task-achieving robots. Motivated by the self-organization abilities of social insects, the authors proposed control mechanisms that allow the system of robots to perform tasks without centralized control or explicit communication. Both simulation and real-robot experiments on box-pushing tasks have shown the feasibility of the approach.

[Mataric1992] discussed the problem of distributing a task over a collection of homogeneous mobile robots. Instead of using hierarchical methods, their approach attempts to detect and use the run-time group dynamic within the robot group. The authors demonstrated through experiments that interference between robots is an important factor in multi-robot control, and achieving linear improvement in performance is non-trivial. However, super-linear improvement in efficiency can be achieved not only by attempting to overcome the effects of robot interferences, but also by explicitly exploiting them to speed up the multi-robot system.

In the next section, another important issue in collective robotics — communication between robots will be discussed.

2.5 Inter-robot Communications

Communication between multiple cooperating robots can improve their capabilities and effectiveness in completing collective tasks. Various approaches for coordinating multiple robots through inter-robot communication have been proposed. For example, [Asama1991] presented a centralized planner that distributes commands to all the robots using a communication system, [Ishida1991] devised a communication protocol for use in robot-to-robot communication system, and [Yin1992] analyzed the performance of Token Bus LAN in coordinating multiple robots. However, global

communication is not practical in large robot groups due to bandwidth limitations. The question is: to what extend can inter-robot communication be used without affecting the overall performance of the entire robot group?

[Balch1994b] investigated the robot system performance of three types of communications on three different robot tasks. The communication types are *No Communication*, *State Communication*, and *Goal Communication*, while the three robot tasks are *Forage*, *Consume*, and *Graze*. The authors argued that the improvement of performance though communication depends on the task, and in cases where communication helps, the lowest level of communication yields the highest relative improvement.

[Yoshida1994] argued that for the cooperation in a large system with many mobile robots, local communication system is considered appropriate in terms of the cost and capacity of communication. However, the behavior of robots has an important effect on the efficiency of communication in such a local communication system. The authors introduced a simple group behavior to improve the communication efficiency. They analyzed the effect of group behavior on the communication performance, and presented a self-organization algorithm for group forming designed for local communication system.

Other researchers, including [Arkin1992] and [Kube1994] proposed decentralized control strategies without explicit communication between robots. Such a decentralized non-communicating system will scale more easily as more robots are added. Also, the system will produce more robust performance in hostile environments. However, the efficiency in achieving collective tasks is often lower in non-communication systems. Also, the sensory requirements for non-communication multi-robot systems are usually higher than that for communication systems.

In the next section the self-organization behavior and formation generating phenomena of multi-robot systems will be discussed.

2.6 Self-Organization and Formation Generation

In achieving collective tasks, sometimes the robots in the group can form certain spatial patterns (formations). These patterns usually emerge from the task achieving behaviors of the individual robots in the group. This property is sometimes called "spatial self-organization" [Unsal1993]. The *formation generation* or *pattern generation* phenomenon is related to formation marching problem that will be discussed in the rest of this thesis, in that a robot formation must be first form before its maintenance becomes an issue. A more detailed discussion of existing research results in formation generation is provided in this section.

[Wang1989] discussed the pattern generation problem in cellular robotic systems(CRS). A CRS employs a large but finite number for robots and operates on a finite n-dimensional cellular space under distributed control, while the pattern generation problem is to design a protocol (a distributed algorithm) to be executed by all robots such that starting from any configuration, the system converges to a given desired pattern in a finite amount of time by moving the objects around within the field of operation. The authors presented protocols that generate patterns in 1-d linear arrangements and protocols that make robots to "seal" one side or all sides of 2-d grids. However, the CRS model is a much simplified model of general multiple robotic systems since the robots are confined on a finite cellular space.

[Sugihara1990] discussed distributed motion coordination of multiple mobile robots. Three problems of coordinating the motion of a group of mobile robots are investigated:

- Form an approximation of a simple geometric pattern, such as a circle, a simple polygon or a line segment;
- Distribute the robots nearly uniformly within a circle or a convex polygon;
- Divide the robots into two or more groups of about the same size.

The authors presented simple and fully distributed solutions for the three problems stated above, assuming that each robot has a sensory device for determining the relative positions of other robots. The basic idea is for each robot to execute a simple algorithm and generate its own schedule adaptively based on the given goal and the positions of certain other robots without explicit communication among them. It is worth noticing that in their approach, they assume that the robots can not determine their absolute positions in the plane, that is, there is no global reference frame for the robot group. It can be easily seen that in many multi-robot applications, for example, space exploration, a global reference frame is either impossible of difficult to obtain, and it would be more reliable for a robot group if the actions of each robot based only on local interactions. The neighbor following approach to be discussed later in this thesis also assumes no global reference frame.

[Genovese1992] discussed self organizing behavior in a distributed robotic system. They tried to interpret the concept of pattern generation as the attainment of a desired equilibrium configuration in proximity of the global goal or, more generally, when a pattern has been recognized to be the one that fits best with a certain working condition. However, their method is not aimed at generating patterns, and the patterns generated by robot units in the group are side effects of the goal-achieving behavior (attracted by pollutants while keeping distance from it and other robots). The patterns are not exact, and only some geometric features are given.

[Unsal1993] discussed spatial self-organization of a group of mobile robots. He proposed distributed algorithms of forming a circle around a goal and uniformly distributing robots inside the circle. His algorithms are similar to that of Sugihara's, but he used a goal behavior to avoid detecting the farthest robot in the group. He also extended the method to 3-d space.

The various approaches discussed in this section address the problem of how to organize multiple robots into a group with certain formations to achieve collective tasks, however, they do not concern how the robot group maintain those formations

while moving around the environment. In the next section, the formation marching problem is discussed.

2.7 Previous Research in Decentralized Formation Marching

In some multi-robot applications such as transporting tasks[Stilwell1994], military operations[Noreils1992, Brock1992], and space missions, it is often desirable for the robots in a group to maintain a pre-specified formation to achieve the collective tasks efficiently and reliably. In this section, some natural phenomena of formation marching, especially the formation flight of various kinds of birds are introduced, then some previous attempts in the formation marching of multiple robots are discussed.

2.7.1 Biological Evidence

Formation marching is a common phenomenon of many kinds of animals, especially birds. Many birds choose to live in close societies. When observed traveling some distance, they often choose to fly in *aggregations* or *flocks*. In [Heppner1974], a flight aggregation is defined as

> "a group of flying birds, lacking coordination in turning, spacing, velocity, flight direction of individual birds and time of takeoff or landing, assembled in a given area,"

where a flight flock

> "is a group of flying birds, coordinated in one or more of the following parameters of flight: turning, spacing, velocity, flight direction of individual birds, and time of takeoff and landing."

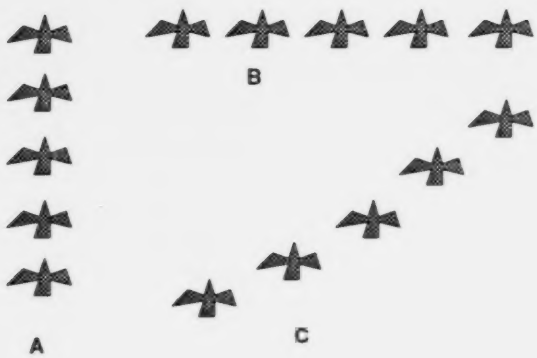


Figure 1: Line formations. A: column; B: front; C: echelon.

Figure 2: Line formations. A: J; B: V.

In [Heppner1974], Heppner described two main types of flight formations, namely, *line* formations and *cluster* formations. Line formations can be further divided into *column*, *front*, *echelons* (see Figure 1), *J* and *V* (see Figure 2), *inverted J* and *inverted V* (see Figure 3), *closed line*, and *compound line* formations (*branched Js* and *branched Vs*) (see Figure 4). Cluster formations can be further divided into *globular cluster*, *front cluster*, and *extended cluster*. For example, the formation most commonly associated with Canada geese (*Branta canadensis*) is the V formation.

Why do birds fly in formation? [Heppner1974] analyzed some possible hypotheses to explain the line formations, especially columns, echelons, and Vs, associated with large birds. One hypothesis is the possible aerodynamic advantages when flying in formations. However, as reported in [Gould1974], the flight formation of birds is not rigidly maintained and aerodynamic hypothesis can not be fully satisfactory.

Figure 3: Line formations. A: inverted J; B: inverted V.

Figure 4: Line and compound formations. A: closed line; B: branched V.

Another hypothesis is that flying in formation enables a bird to see both the individual in front of it and obtain a clear view forward. When [Williams1976] used radar to measure the angles of Canada goose flight formations, they concluded that formation flying probably has behavioral advantages rather than simple aerodynamic ones. [Heppner1985] investigated the relation between of visual angle and formation flight in Canada geese, and suggested that since many birds have central monofovea, it would be helpful to align oneself in the formation such that a neighbor ahead would be positioned on one's optic axis.

Yet other hypotheses exist. [Ward1979] suggested that formations may serve as a signal function such that birds can recognize their own species at a distance. This ability would allow geese to join flocks already in migration. [Pomeroy1992] studied the turning structure in airborne rock dove (*Columba livia*) flocks and suggested formation flight may be used to detect possible predators.

It is also found that the individuals in a flight formation reposition themselves frequently. For example, the leader bird of a goose flock may occasionally give up its position to another bird. [Heppner1974] suggested that this may be owing to energy preservation reasons or stimulus fatigue.

In conclusion, it appears that scientists are still searching for the answer to why some birds fly in formation. However, one characteristic of the formation flight of large birds can be drawn from the above observations; that is, each follower bird tries to maintain a fixed distance and angle with one or several of its nearby birds while flying. The neighbor following approach for formation marching of robots that will be introduced in the next chapter generally follows this idea. In the next section some previous approaches to control a group of mobile robots moving in formation will be examined.

2.7.2 Previous Research

In a formation marching problem, a group of robots are required to move to a certain goal position while trying to maintain the formation of the group. So, a formation marching problem is a formation keeping problem. Because formation marching of multiple robots has a large variety of applications in military operations, space missions, and industry automation, it has raised the interests of many researchers. In this section, several previously proposed approaches towards this problem will be discussed.

[Wang1991] proposed several distributed navigation strategies for robots moving in formation. Each robot is treated as a mass point (thus no holonomial constraints) and all the strategies assume a world coordinate frame (world frame) is known by all robots in the group (fleet), so each robot knows the absolute coordinates of one or more nearest neighbors in the group (*nearest neighbor tracking* and *multi-neighbor tracking*, accordingly) and adjusts its movement accordingly. Up to 4 robots moving in complex trajectories while keeping various formations are shown in simulation.

[Chen1994a, Chen1994b] also investigated formation generation and formation marching of a small group of homogeneous mobile robots by imposing constraints. In pattern generation, a modified version of [Sugihara1990]'s algorithm is presented to avoid collision and provide better formation. In formation marching, they assume each robot has the knowledge of its current position and orientation with respect to the world coordinates. In addition, the leader robot of the group broadcasts its position and orientation with respect to the world coordinate frame to all other robots in the group. In [Chen1994b], two types of formation strategies that have different properties in maintaining the orientation of the overall formations are implemented using different algorithms, and the algorithms are verified by simulation showing the coordinated motion of robots moving in square and circular formations.

In the above approaches, a substantial amount of global information including world coordinates are required. There are also formation strategies mainly based on local interaction of robots in the group. [Parker1993] investigated the performance of formation marching under a single leader. The robots in the group have the following behaviors: *Move-to-Goal*, *Keep-Formation*, and *Avoid-Obstacles*. The Keep-Formation behavior requires a group of robots to stay in formation with one another (i.e. remain aligned side by side) while the leader of the group follows a prespecified route. Each robot can sense the location of its neighboring robots relative to itself (local knowledge) and is physically constrained by the inability to move backwards. The author discussed how the performance of the robot group is affected by the different proportion of global and local control information. She argued that to get satisfactory results, the following robots should have a proper balance between local and global control.

[Balch1994a] discussed motor schema-based control for multi-agent robot formations. He considered four formations for a group of four robots: *line*, *column*, *diamond*, and *wedge*. For each formation, each robot has a specific position and is assigned a unique ID. The *maintain-formation* motor schema generates a movement vector toward the desired formation position by using artificial potential fields. Three

different formation position references are introduced: *Unit-center referenced*, *Leader-referenced*, and *Individual-referenced*, However, only the simulation results of the unit-center reference method is available.

[Brock1992] discussed the coordination and control of multiple autonomous vehicles in the DARPA SIMNET project. The DARPA SIMNET project allows hundreds of soldiers to train together in a virtual air, land and sea environment through a network of interactive simulators. In addition to the manned simulators, the virtual environment also contains a large number of autonomous vehicles called Semi-Automated Forces (SAF) which are coordinated by an operator at a single workstation. The SAF vehicles are simulated mobile robots operating in a complex environment with other autonomous vehicles and manned simulators. The SAF vehicles accept high level commands from a supervisor and are responsible for lower level control such as obstacle avoidance, formation keeping, and path planning.

In [Brock1992], a group of SAF vehicles has a leader vehicle that is usually located near the head of the group. Each member of the group has the knowledge of the relative position and heading of the leader and has a station point representing the desired position of the vehicle relative to the leader. The objective is to stay in formation while maneuvering among the obstacles and other robots. Formation motion is obtained by projecting the follower's station point forward in the direction of the leader's intended motion. Motion is then allowed only if the follower is either a significant distance from the group or its velocity is in the same direction as the leader's. By scaling the velocity proportional to the distance from the projected point, the vehicle will either speed up or slow down to fall in line with the formation. Simulation shows that this algorithm can achieve satisfactory formation performance.

Obviously there are trade-offs between using centralized and decentralized control methods. To coordinate multiple robots moving in formation without central coordination, position information can be acquired by either sensing or explicit inter-robot communication. However, as we have stated, extensive use of communication leads

to communication bottlenecks or loss of information, and inter-robot communication is not always possible or reliable. In contrast, decentralized coordination of a group of robots moving in formation tends to reduce the need for communication, although the difficulty now shifts to sensing systems of the robots.

2.8 Summary

In this chapter, the concept of behavior-based robotics is first introduced. Both Brooks' subsumption architecture and Arkin's motor schema based reactive control are discussed. It has been shown that behavior-based control is a promising way to design autonomous intelligent robots that are situated in the real world. How multiple behavior-based mobile robots can be organized into task-achieving groups by inciting collective behaviors is shown next. By studying the group behaviors of social insects and other animals, many researchers have proposed different approaches to the control of robot groups to achieve various tasks. Then two important issues in controlling collective robots – the control architecture for the robot group and inter-robot communications – are discussed. It has been argued that decentralized control approaches are more suitable for large robot groups, and excessive or global explicit communication should be avoided, especially for large robot groups. Also, pattern formation of multiple robots which is a related topic to the formation marching problem being investigated in this thesis is discussed. Finally, how birds coordinate their motion while flying in formation are presented and some previously proposed approach to the formation marching of multiple robots are investigated.

In the next chapter, the neighbor following approach towards the decentralized control of multiple robots moving in formation will be presented.

Chapter 3

The Neighbor Following Approach

3.1 Introduction

Motivated by the formation flight of birds, previous research on formation marching of multiple robots, and vehicle-following systems [Tsumura1992], a decentralized control strategy for multiple robots moving in formation, namely, the *neighbor following* approach, is developed in this thesis and will be discussed starting from this chapter.

This chapter is organized as follows: First, some notions used throughout the rest of this thesis are introduced. Second, the design considerations and advantages of the neighbor following approach are presented. Third, the robot model used in this research is presented. Then, the single neighbor following strategy is introduced, followed by the multi-neighbor following strategy. Emphasis will be put on the algorithms of how to maintain arbitrary formations, reduction of formation error through motion prediction, and speed and orientation estimation of a leader robot through position measurements. Also, the requirements for the order in which the robots follow each other are discussed. The implementation details will be discussed in Chapter 4, and simulation results will be presented in Chapter 5.

3.2 The Neighbor Following Approach to the Formation Marching of Multiple Robots

3.2.1 Definitions

Our discussion begins with the definitions of terminologies that will be used in describing the neighbor following approach:

- *Formation*: A formation is the pre-defined geometry pattern of a group of mobile robots to be maintained while doing collective tasks;
- *Leader*: In this thesis, the concept of leader is relative. A leader robot A of another robot B means that the motion of B is entirely or partially depends on the motion of A. A leader robot itself may be a group leader or a follower robot.
- *Group leader*: A group leader is a robot whose motion is independent of the motions of other robots in the robot group except for collision avoidance purpose. A group leader is responsible for its own path planning. Alternatively, it may receive navigational commands from an external supervisor, and its motion directly affects the collective motion of the whole robot group. There can be one or multiple group leader robots in a robot group;
- *Follower*: A follower robot is a robot whose motion is dependent of the motions of one or more other robots (i.e., its leaders). A follower robot itself can be a leader of other robots.

Figure 5 shows a triangle formation of 6 robots to clarify the above definitions. In this sample formation, robot 1 is the group leader and is the leader of robot 2 and 3. Robot 2 and 3 are follower robots of robot 1, but leader robots of robot 4 to 6. Robot 2, 3, 4, and 6 follow only one leader robot each, but robot 5 follows both robot 2 and robot 3.

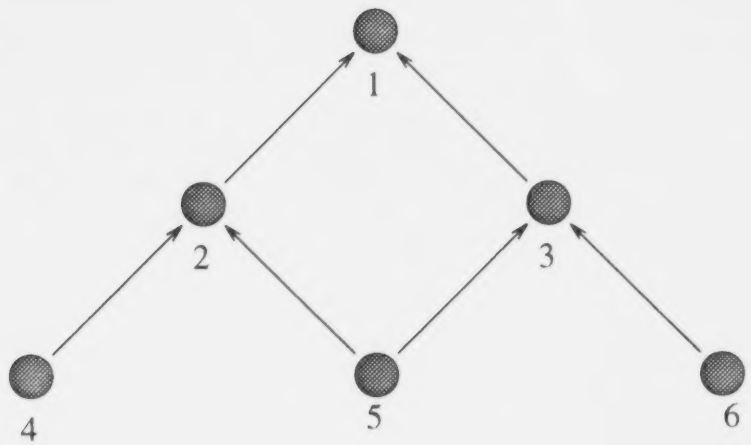


Figure 5: A simple formation of 6 robots.

3.2.2 What is Neighbor Following?

In order for a group of mobile robots to move in formation, each robot in the group must maintain a specified position and orientation with respect to one or several reference points that will determine where the robot should be in the formation. As [Balch1994a] pointed out, this reference point can be either the group center (*group center following*), the navigational leader robot(s) of the group (*leader following*), or one or several other robots in the group, not necessarily the navigational leader(s) of the group (*neighbor following* or *individual following*). However, group center following is not suitable for distributed control of large robot groups because if each robot has to compute the position of the group center, it has to sense and average the x and y positions for all robots in the group with respect to one particular coordinate frame, so the computation will be expensive. In addition, due to limited sensing range and possible occlusion, it will be almost impossible if the number of robots in the group is large. Leader following approach imposes similar problems. When the robot group is large, the navigational leader can not always be seen by each follower robot because of possible occlusions. If the leader is far away from a follower robot, the position and orientation of the leader may either be inaccessible by the follower because of limited sensing range or inaccurate because of sensing error.

In contrast to the group center following and leader following approach discussed above, the neighbor following approach has several advantages in decentralized control for large robot groups including

- Each follower robot only needs to know the location and direction of several nearby robots instead of all other robots in the group; hence sensing and computation will not become prohibitively expensive for large robot groups;
- The effective sensing distance can be smaller because a follower robot need not sense the location and orientation of a far away leader robot;
- Occlusion will be less a problem because there will be fewer objects between the following robot and a nearby leader robot.

Neighbor following certainly has some problems such as error propagations caused by complicated following hierarchy, but by carefully arranging the following sequence, the effect of error propagation can be minimized.

The concept of using neighbor following to maintain formation of a group of mobile robots has been proposed by some researchers. For example, [Wang1991] proposed using *nearest neighbor tracking* to control a fleet of mobile robots. However, in his approach, the robots are treated as points, and a world coordinate frame is used. In the neighbor following approach, there is no world frame for the robot group, and each robot only reacts to the world using its sensory information. Also, physical constraints of the robots are taken into consideration. The *individual reference* technique introduced by [Balch1994a] is also similar to our method, but the neighbor following approach allows a follower robot to have more than one leader to coordinate its motion to produce more robust performance.

Before introducing the neighbor following algorithms, the robot model that will be used throughout this thesis is first described.

3.2.3 The Robot Model

Here we describe the robot model used in our investigation of the neighbor following approach to control the formation marching of multiple robots:

A. Structural Properties

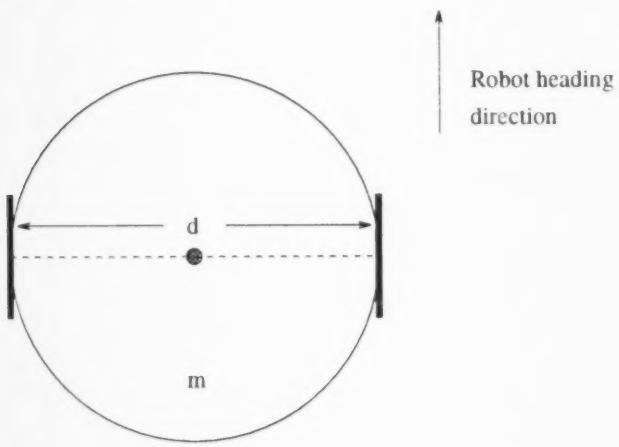


Figure 6: Physical structure of a mobile robot.

Each robot is equipped with two wheels that are driven separately (see Figure 6). It is circularly shaped with diameter d and mass m. The robot's two equal-sized wheels are placed on both sides of the robot. The distance between the two wheels is also d and the line connecting the two wheels passes through the center of the circle of the robot, which is also the center of mass of the robot. The robot has two degrees of freedom, i.e., it can move along its heading direction and rotate about the axis normal to the ground but can not move sideways. The speed of the two wheels has an upper bound which in turn defines upper bounds on both the linear velocity and angular velocity of the robot. The mass of the robot is considered in the calculation of robot acceleration.

B. Perception Abilities

Each robot has many different sensors that enable it to sense the existence of nearby robots and obstacles. A leader robot of the robot group has goal sensors that enable it to navigate towards a goal position or move along a specified path while a follower robot only needs to maintain a pre-defined location with its leader robots.

The sensing abilities of each robot is limited in the sense that:

1. The perception range of the robot has a limit, that is, it can not sense objects beyond a certain distance;
2. The robot does not have an external coordinate frame. This means that the robot can only depend on its own current coordinate frame;
3. Sensors are inherently noisy.

In our neighbor following approach, we assume each robot except the group leader(s) has the following knowledge of any robot it is following (see Figure 7):

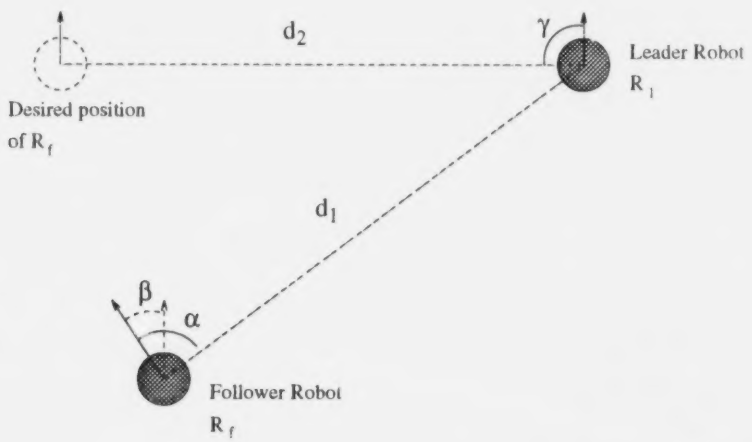


Figure 7: Spatial Information needed for a robot R_f to keep a specific position with respect to another robot R_l.

1. Distance between the two robots d_1;

2. Relative angle α of the tracked robot with respect to the following robot's reference frame;

3. Relative angle β of the heading direction of the tracked robot with respect to the follower robot's reference frame;

4. Desired distance with the leader robot d_2;

5. Desired angle γ in the frame of the leader robot.

The first two items of information can be obtained by the follower robot via sensor arrays. The last two items of information are usually built-in knowledge of the follower robot. The relative heading direction of the leader robot α can be obtained by inter-robot communication or can be estimated by the measuring the change in position of the leader robot which will be discussed later.

C. Communication Abilities

The robot only has a limited ability of explicit local communication. The activities of the robot mainly depend on its perception of the environment. Each robot can obtain its own orientation using a built-in compass, but it can not determine its location in any external coordinate frame. Each robot can use local communication to send its own orientation and speed to nearby robots as well as to receive the orientation and speed of other nearby robots. Formations can also be maintained without explicit inter-robot communication, as will be discussed later in this chapter, but it will put a much higher demand on the sensor system.

D. Computation Abilities

Each robot is driven by the combined effect of several *behaviors*. The behaviors are more like Arkin's motor schema [Arkin1987] than Brooks' subsumption architecture [Brooks1986] in the sense that each behavior produces a force vector and the motion

of the robot is according to the normalized vector summation of all the force vectors produced by the individual behaviors. Besides the *keep formation* behavior for the non-leader robots, each robot has an *avoid obstacle* behavior, and each group leader has a *goal behavior* to navigate it towards a goal position or through a specified path.

3.2.4 Single Neighbor Following Algorithm

Two formation keeping strategies are proposed in this thesis, namely, *single neighbor following* and *multiple neighbor following*. In single neighbor following, there is one group leader moving on its own and each follower robot tries to keep a pre-specified position and orientation with respect to its assigned leader robot. Each follower robot has a *leader robot*, which is not necessarily the group leader. In multiple neighbor following, however, there can be more than one group leader navigating on their own, and each follower robot can follow more than one leader robot. We will first introduce the formation keeping algorithm for single neighbor following, and then extend the algorithm to multiple neighbor following.

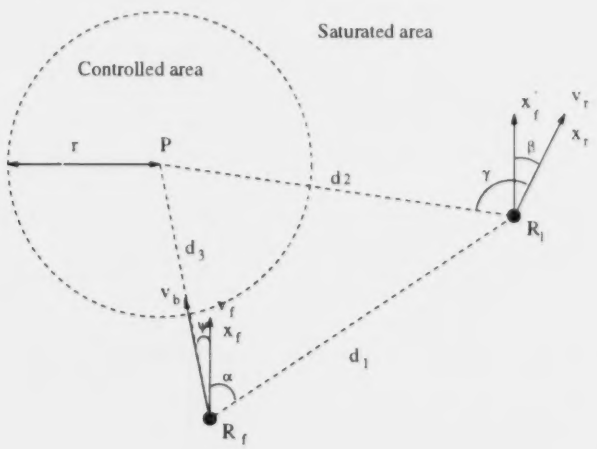


Figure 8: Calculate follow behavior of a follower robot

The force vector produced by the keep-formation behavior of each follower robot comes from an artificial potential field. Specifically, any robot R_f in the group that is not the group leader calculates the position and heading of its leading robot R_l with

respect to R_f's reference frame. It then generates an artificial potential field around the desired formation position of R_f with respect to R_l. The attractive potential in turn guides R_f to move to its desired position in the formation, subject to kinematic and dynamic constraints defined in the previous section.

Figure 8 illustrates how the keep formation behavior for one leader robot in 2-D space is calculated. At a given instant, R_f determines from the sensor information its reference robot R_l to be at a distance d_1 and a relative angle α. In addition, assume the angle between the moving direction of R_r and that of R_f to be β. According to the reference information known by the follower robot through communication or sensor measurements , the follower robot should be at point P of distance d_2 at an angle γ with respect to R_l. In order for R_f to reach P, the keeping formation algorithm needs to know the distance d_3 between the current position of R_f and P, and the angle ψ between the current direction of R_f and the vector $\vec{v_b}$ from R_f to P.

Using simple transformation matrix calculation, we can obtain the position of $P(x, y)$ in the follower robot frame as follows:

$$x = c\cos\beta - d\sin\beta + a \tag{3.1}$$

$$y = c\sin\beta + d\cos\beta + b \tag{3.2}$$

where

$$a = d_1\cos\alpha \tag{3.3}$$

$$b = d_1\sin\alpha \tag{3.4}$$

$$c = d_2\cos\gamma \tag{3.5}$$

$$d = d_2\sin\gamma. \tag{3.6}$$

The above equations can be simplified as:

$$x = d_2\cos(\beta+\gamma) + d_1\cos\alpha \tag{3.7}$$

$$y = d_2\sin(\beta+\gamma) + d_1\sin\alpha \tag{3.8}$$

d_3 and ψ can then be defined in terms of x and y as follows:

$$d_3 = \sqrt{x^2 + y^2} \tag{3.9}$$

$$\psi = \arctan \frac{y}{x} \tag{3.10}$$

Once d_3 and ψ are known, the keep formation behavior generates a force vector $\vec{F}_b$ towards P. The magnitude of $\vec{F}_b$ depends on d_3. As the velocities and accelerations of the robot have upper limits, the force vector $\vec{F}_b$ must also have an upper limit. This upper limit is defined as F_{max}. We define an area within radius r of the desired position P as the *controlled area*, and the area outside this area *saturated area*. When robot R_f is in the controlled area, the magnitude of $\vec{F}_b$ is proportional to d_3. Otherwise, the magnitude of $\vec{F}_b$ is equal to F_{max}. The force function we use thus has the form

$$\|\vec{F}_b\| = \begin{cases} \frac{d_3}{r} F_{max} & 0 \leq d_3 < r \\ F_{max} & d_3 \geq r \end{cases} \tag{3.11}$$

3.2.5 Eliminate Static Formation Error by Dynamic Prediction

When using the single neighbor following algorithm discussed in the previous section, static position errors in the formation will exist in order to activate the force function to move the robots in the formation. In a decentralized control scheme, if they are left uncompensated, these errors will propagate through the formation and degrade the formation quality, a phenomenon that is particularly pronounced in large robot groups. Fortunately, this problem can be overcome by moving the following robot at the same speed and direction as its leader, even in the absence of position errors. If a following robot knows the accurate relative position and orientation of its leading robot all the time, elimination of the static errors can be achieved by putting the attractive center of the force function somewhere ahead of its desired position. The new attractive center must be chosen so that even if the following robot is in its required position, it still keeps the same direction and speed as its leading robot.

Define d_c as the amount by which the new desired position is ahead of the robot's required position in the formation. Apparently, d_c can not be a constant because any robot in a formation might move at any possible speed. Rather than invoking elaborate dynamics-based modeling, the computation of d_c has been based on empirical observations. First, since the following robot must maintain the same speed and direction as its leading robot while in the required position, d_c must be proportional to the speed of its leading robot v_l. Second, it must also be proportional to the mass of the following robot m_f, and inversely proportional to the interval between two consecutive positions Δt. Finally, we should also take into account the maximum force produced by the formation keeping algorithm F_{max}, and the maximum controlled distance d_{max}. Incorporating all the above considerations, we calculate d_c by

$$d_c = K_c v_l \tag{3.12}$$

where

$$K_c = \frac{d_{max} m}{F_{max} \Delta t} \tag{3.13}$$

The x- and y-coordinates of the new attraction center will then be:

$$x = d_2 \cos(\beta + \gamma) + d_1 \cos\alpha + d_c \cos\beta \tag{3.14}$$

$$y = d_2 \sin(\beta + \gamma) + d_1 \sin\alpha + d_c \sin\beta \tag{3.15}$$

We call this method *dynamic prediction*. When the neighbor following algorithm is viewed as a control system that controls the robot to its desired formation position, it can be seen that using dynamic prediction is equivalent to add an integral component to the proportional feedback control loop. With dynamic prediction, we have been able to reduce the static tracking errors between robots substantially. The position error in the formation is therefore by and large contained. This enables our formation keeping algorithm to work well for large robot groups performing complex formations.

Figure 9 and Figure 10 are simulation snapshots of a group of 3 robots moving in a column formation [1]. The following sequences of the robot groups in the two simulation

[1] The simulations are performed using RoboSquad V2.0 which will be introduced in detail in chapter 4.

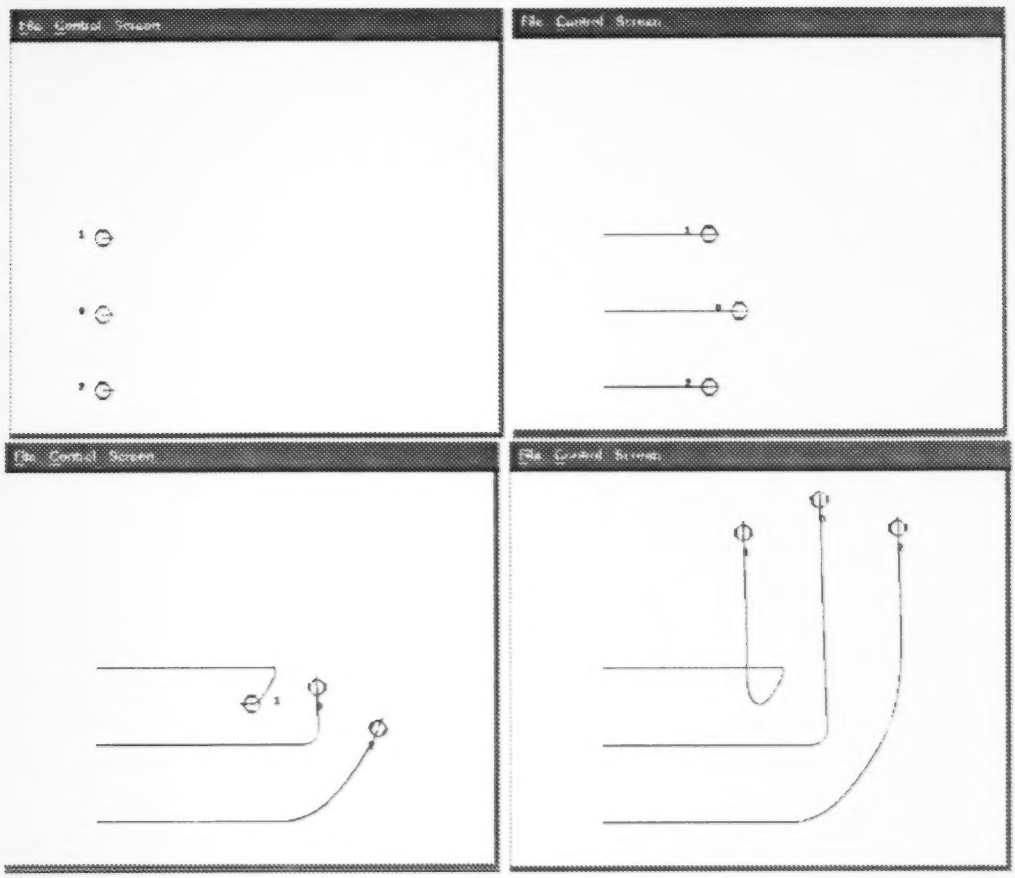


Figure 9: Simulation snapshots of 3 robots moving in a column formation without using dynamic prediction. Upper left: step 0; Upper right: step 172; Lower left: step 351; Lower right: step 598.

experiments are the same (robot 0 is the group leader, and robot 1 and robot 2 follow robot 0) and the robot group moves along the same path (robot 0 makes a left turn of 90 degree). However, in Figure 10, the follower robots use dynamic prediction to overcome static formation error while in Figure 9 the follower robots do not. We can see that, without dynamic prediction, the follower robots will fall behind its desired formation position with respect to its leader robot once the group begin to move and stablize at a distance that is proportional to the leader's speed. On the other hand, if dynamic prediction is used, the static formation errors are eliminated when the leader robot moves in a straight line. In cases where the trajectory of the leader robot is not

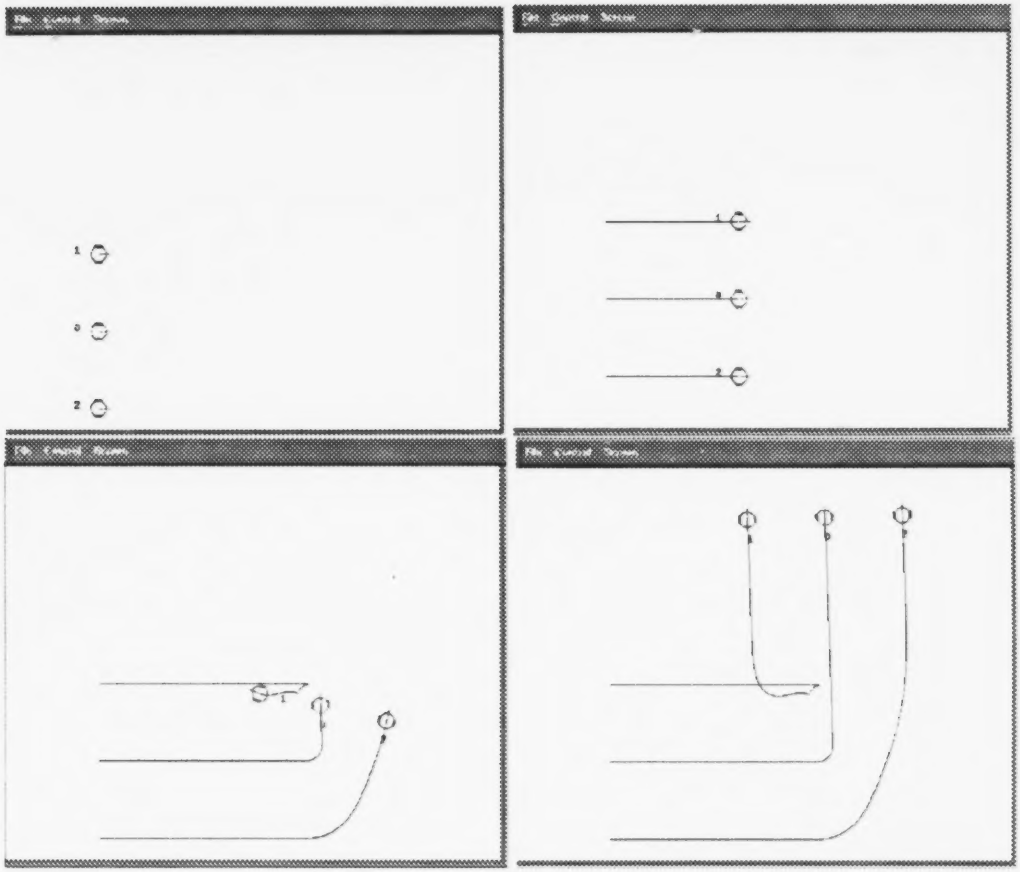


Figure 10: Simulation snapshots of 3 robots moving in a column formation using dynamic prediction. Upper left: step 0; Upper right: step 172; Lower left: step 351; Lower right: step 598.

abrupt, dynamic prediction can effectively reduce static formation error. However, dynamic prediction requires the speed of the leader robot v_l to be accurately known because the seed error in v_l will be enlarged K_c times, and a large seed error causes larges in the formation.

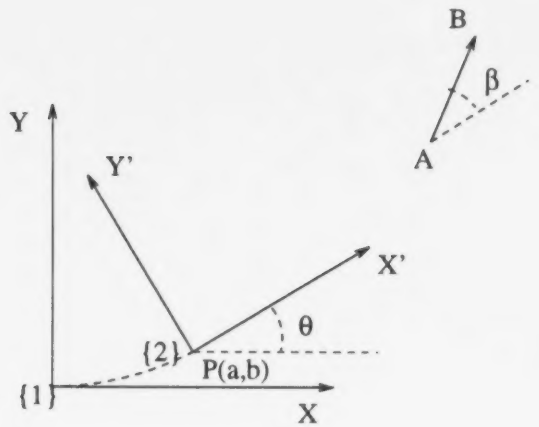


Figure 11: Direction and speed estimation using two position measurements

3.2.6 Leader Direction and Speed Estimation Using Position Measurements

In the neighbor following algorithm, each follower robot must know not only the relative position but also the relative orientation and moving speed of the robot being followed. The latter can be obtained by sensing the heading directly or explicit inter-robot communication. However, the approximate heading and speed of a robot can also be estimated from two or more consecutive measurements of its position, provided the sensors for detecting robots are accurate enough, thus eliminating the need for inter-robot communication and that for additional sensors of instantaneous heading.

Figure 11 shows how the moving direction and speed of a leader robot is measured using two consecutive position measurements. From Figure 11 we can see that at time t_1, the leader robot R_l is at point $A(x_1, y_1)$ with respect to the follower robot R_f's current coordinate frame $\{1\}$. At $t_2 = t_1 + \Delta t$, robot R_f moves to position $P(a, b)$ with an orientation change θ with respect to $\{1\}$, and R_l is now at point $B(x_2, y_2)$ with respect to R_f's current coordinate frame $\{2\}$. Assuming robot R_l's trajectory is a straight line from A to B, to calculate the relative direction θ of vector $\vec{AB}$ with respect to $\{2\}$, we need to know the position change (a, b) and orientation change

α of robot R_f between t_1 and t_2. This can be obtained by dead-reckoning. If Δt is small, we can assume the trajectory of R_f to be either a straight line or an arc, so that (a, b) and α can be estimated by how much the driving wheels of the robot have traveled.

Now we need to represent point A in coordinate frame $\{2\}$. We define, in frame $\{2\}$, point A to be $A(x_1', y_1')$. Using transformation, we have

$$\begin{pmatrix} x_1' \\ y_1' \\ 1 \end{pmatrix} = Rot(-\theta)Trans(-a, -b) \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \tag{3.16}$$

where:

$$Rot(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.17}$$

$$Trans(a, b) = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \tag{3.18}$$

Solving the above equations, we can get:

$$x_1' = x_1 \cos\theta + y_1 \sin\theta - (a\cos\theta + b\cos\theta) \tag{3.19}$$

$$y_1' = -x_1 \sin\theta + y_1 \cos\theta + (a\sin\theta - b\cos\theta) \tag{3.20}$$

Thus, the relative direction β of vector $\vec{AB}$ with respect to $\{2\}$ is:

$$\beta = \arctan \frac{y_2 - y_1'}{x_2 - x_1'} \tag{3.21}$$

and the moving speed of R_l is:

$$v_f = \sqrt{(x_2 - x_1')^2 + (y_2 - y_1')^2} \tag{3.22}$$

Note that the above equation may not be accurate when both $y_2 - y_1'$ and $x_2 - x_1'$ are small. In such cases we assume that robot R_l is not moving at all, so the relative

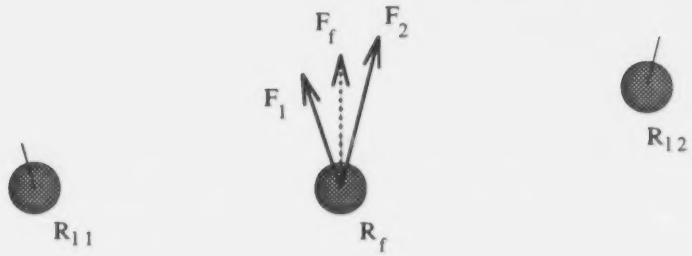


Figure 12: A follower robot R_f decides its motion based on the motions of its two leaders R_{l1} and R_{l2}. F_1 and F_2 are the formation forces produced by formation keeping algorithms associated with F_1 and F_2, respectively, and F_f is the resultant formation force.

orientation of R_l between t_1 and t_2 is considered to be unchanged. When using this strategy, although robot R_f can not detect the relative orientation β of robot R_l if R_l is rotating without position change, it can get correct β as soon as R_l starts moving.

3.3 Multiple Neighbor Following

The principle of multiple neighbor following method is straightforward: instead of using just one robot as its leader, each follower robot uses more than one robot to determine its motion. The aim of using multiple leaders is to try to cancel or reduce possible undesired motion errors of the formation produced by random sensory errors and other behaviors such as obstacle avoidance.

Assume R_f is a follower robot, and $R_1, \ldots, R_n$ are its leader robots. We apply the single neighbor following algorithm on each of the n leaders, where each leader robot $R_i (i = 1 \cdots n)$ produces a formation force $\vec{F_i} (i = 1 \cdots n)$, with a weight of W_{fi}, and compute the total formation force $\vec{F_f}$ by:

$$\vec{F_f} = \frac{1}{n} \sum_{1}^{n} W_{fi} \vec{F_i}$$

In other words, the formation force of a follower robot is the weighted average of the formation forces produced by each of its leader robots. Figure 12 shows how a robot with two leader robots decides its motion. It is worth noticing that if the error

representation of the formation force vector produced by each individual leader robot can be known in advance, then some "optimal" fusion methods such as *maximum likelihood estimation* can be used to provide better error elimination. However, since error in force vector can be caused by many factors, a constant error co-variance matrix for each force vector is difficult to obtain. Fortunately, the weighted average method which is used here can also improve formation performance in simulation experiments as will be shown in Chapter 5.

Multiple neighbor following can also produce more natural and coordinated motion formation for robot groups moving through various obstacles. Since many follower robots now determine their motions from more than one leader robots, occlusion and obstacle avoidance actions of any of their leaders will have less effect on the motion of the follower robot. However, the weighted average method is not suitable for the performance measurement of the robot group here, and a proper performance criterion is yet to be defined as will be further discussed in Chapter 5.

3.4 Neighbor Following Sequence

When neighbor following approach is used to control a group of mobile robots moving in formation, the neighbor following of a group must have a partial order; that is, if a robot R_l is followed by a robot R_f, it can not follow R_f either directly or indirectly. So, in single neighbor following, the overall following architecture should be a tree, while in multiple neighbor following, the following architecture is a partially ordered network. The reason for this partial order is to minimize the disturbance of the whole formation caused by the formation error or undesired motion of each individual robot in the group. Figure 13 gives two examples corresponding to single neighbor following and multiple neighbor following, respectively.

It is worth noticing that to achieve better formation performance, the depth of the following architecture, i.e., the depth of the tree in single neighbor following and

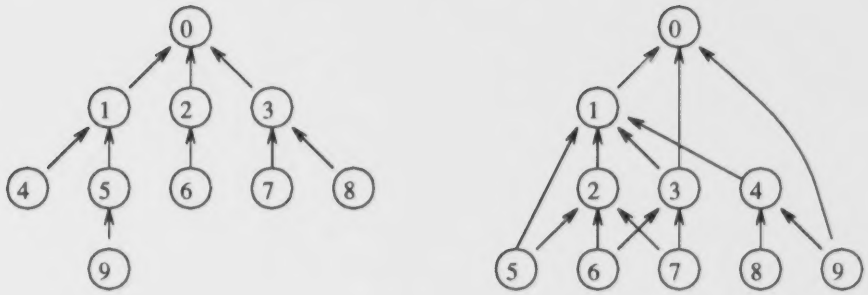


Figure 13: Different Neighbor following architectures. Each numbered circle in the figure represents a robot with a different ID, and each arrow represents a follow relationship from a robot to another robot. A: single neighbor following; B: multiple neighbor following

the depth of the network in multiple neighbor following should be as small as possible to reduce the effect of error propagation. However, the distance between a follower robot and its leader robot should also be as small as possible because the sensed leader position error produced by sensor angle error is proportional to the distance between the two robots. When there exists conflict between the above two requirements, a trade-off must be made.

3.5 Summary

In this chapter, the decentralized control for multiple robots moving in formation – the formation marching problem is presented. Two types of neighbor following are covered, namely, single neighbor following and multiple neighbor following. Methods to enhance the neighbor following algorithms, i.e., how to estimate another robot's heading direction by position measurements, and how to minimize formation error by using dynamic prediction are also discussed. Finally, the requirements of how the robots follow each other in implementing the neighbor following algorithms are described.

In the next chapter, the robot population simulator for the formation marching

problem – *RoboSquad* will be introduced. Its architecture including models for sensors behaviors, behavior integration, and actuators is discussed. Facilities for testing the neighbor following algorithms are explained.

Chapter 4

Implementation

4.1 Introduction

The final goal of this work is to design and build physical robots capable of maintaining pre-defined geometric patterns while moving around the environment. However, when developing control model for multi-robot systems, simulation is usually first used to test various protocols until a control model that seems feasible in physical robots is developed. As [Kube1992b] pointed out, computer simulation as a powerful tool in the early developing stages of a multi-robot system has several advantages such as easy to build, change, and reconfigure both the test environment and the control models. Simulation is also more cost effective since less early investments are required in the development process. However, in simulation, unrealistic simplifications and assumptions made in modeling the robot system or the environment may cause the control model unusable in real robots later on. Fortunately, this problem can be minimized if we avoid unrealistic assumptions, treat simulation just as a tool for testing the feasibility of our control theory, and test the theory in real robots eventually.

In this chapter, the robot population simulator *RoboSquad* which is used in the study of formation marching of multiple robots is presented. First, the architecture

of RoboSquad is introduced, Second, the models for sensors, behaviors, behavior integration, and actuators are discussed. Finally the features provided by user interface of RoboSquad are briefly presented.

4.2 The Architecture of RoboSquad

The RoboSquad simulator consists of three parts, namely, *robot*, *environment*, and *user interface*. The environment part of the simulator is responsible for creating the simulated environment with various obstacles in which the robot group navigates. The robot part is responsible for controlling the robots to move around the simulated environment, and the user interface part is responsible for the graphic display of the simulated environment with multiple robots moving around. The user interface also provides a means for the user to issue control commands and gather test results. The structure of the simulator and the interaction between its three components are shown in Figure 14.

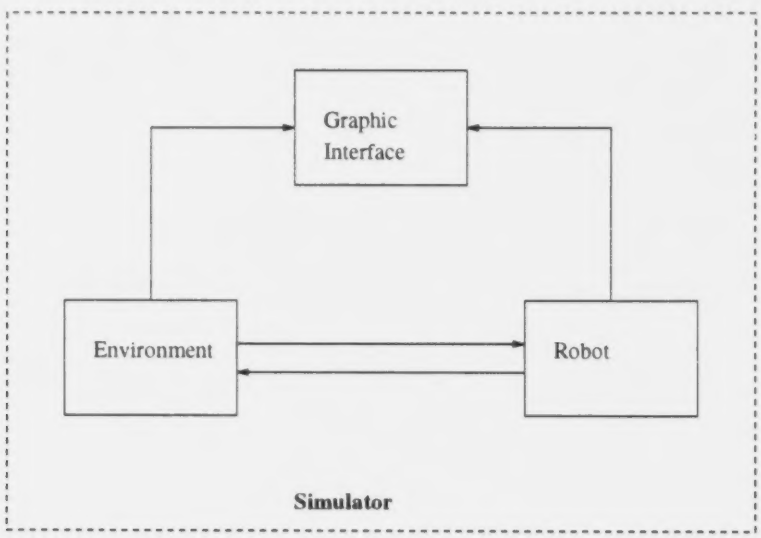


Figure 14: Structure of RoboSquad

The concept of object oriented design is extensively used in designing RoboSquad.

For example, the environment part consists of two types of objects:

- 2-dimensional Cartesian coordinate system;
- Obstacles that can be distinguished by their different properties such as their position in the coordinate frame, and the number of sides.

In addition, the initial positions and status of robots, the path for group leaders, the locations of obstacles, and the appearance of the user interface are all reconfigurable by using configuration files.

4.3 Robot Modeling and Implementation

The robot part of RoboSquad consists of a group of simulated mobile robots. Each robot object is assigned an unique identification number (ID). A robot's position in formation depends on its ID. Each robot can either be a *group leader* or a *follower* robot. Each robot has several *sensors*, a *controller* and a pair of *actuators*. Also, each robot has a small memory system that enables it to keep track of its own position and direction changes and the position changes of its leader robots. The structure of a robot is shown in Figure 15.

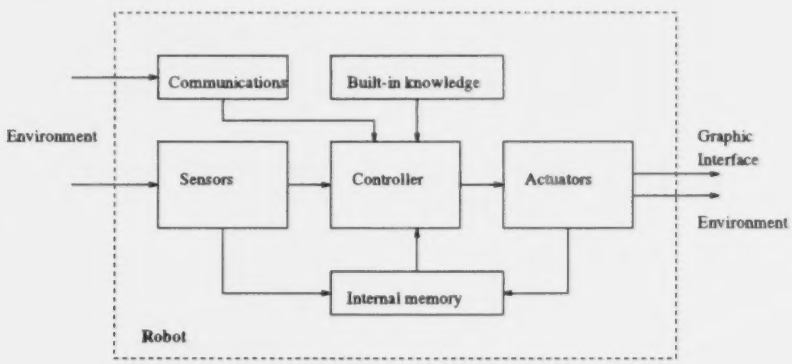


Figure 15: Structure of the robot part of RoboSquad

4.3.1 Sensors

There are two types of sensors in each follower robot – *obstacle sensors* and *robot sensors*. The obstacle sensors are responsible for detecting nearby obstacles (including other robots), and robot sensors are responsible for detecting the robots it is following. Since in the current version of RoboSquad, the navigation information of each group leader is given through a configuration file, a group leader robot only has obstacle sensors.

Each follower robot has one or several robot sensors, and each robot sensor is responsible for detecting the distance and relative angle of a leader robot. If the robot is within a certain distance, the robot sensor is activated and generates distance and relative angle values; otherwise, it is deactivated. The output values of the robot sensor may have normal distributed distance error and angle error with given variances. The data structure of a robot sensor is shown below:

```
struct RobotSensor {
        int activated;          /* Set to 1 if sensor is activated */
        int robot_id;           /* ID of the robot */
        float distance;         /* Distance of the robot */
        float angle;            /* Relative angle of the robot */
        float disterror;        /* Variance of distance error */
        float anglerror;        /* Variance of angle error */
};
```

Each robot also has a number of obstacle sensors that enable the robot to detect the existence of nearby obstacles. Only objects that fall in a certain distance will be detected. The obstacle sensors do not discern real obstacles with other robots in the group. When activated, the obstacle sensor gives out values of distance and relative angle of the nearest point of an obstacle. The output values of the obstacle sensor may also have normal distributed distance and angle error with given variances. The

obstacle sensor model is as follows:

```
struct ObstacleSensor {
        int activated;          /* Set to 1 if the sensor is activated */
        int obstacle_id;        /* Assign an ID to the obstacle */
        float distance;         /* Distance of the obstacle */
        float angle;            /* Relative angle of the obstacle */
        float disterror;        /* Variance of distance error */
        float anglerror;        /* Variance of angle error */
};
```

4.3.2 Behaviors

In the current version of RoboSquad (Version 2.00), each behavior accepts data from sensors and produces a force vector to guide the motion of the robot. Also, each behavior is assigned a weight to represent its effect on the resultant motion of the robot. The behavior model is as follows:

```
struct Behavior {
        float weight;
        float output_force;
        float direction;
};
```

Three behaviors are implemented, namely, *goal* behavior for group leaders, *formation keeping* behavior for follower robots, and *obstacle avoidance* behavior for all robots. These behaviors enable the robot group to navigate in formation through an environment possibly with obstacles existing.

A. Goal Behavior

The goal behavior of a group leader robot is responsible for navigating the robot to one or several goal positions or along a pre-specified path. Currently, three control strategies are implemented, namely, *speed–time*, *position–time*, and *position–speed*. The speed–time method guides the robot to move at a constant velocity (both linear and angular) for some time, before the robot accepts a new velocity and time. This method is ideal for testing the formation keeping behavior in obstacle-free environment without using the obstacle avoidance behavior since the trajectory of the group leader robot can be precisely defined. However, when obstacle avoidance behavior is activated, the leader robot may deviate from its pre-assigned path. The position–time method guides the group leader to move to a pre-defined position in a given time, when this position is reached, a new position and time can be given. The position–speed method guides the group leader robot to move to a pre-defined position at a nearly constant velocity. When the robot reached the position, a new position and velocity can be given. The latter two methods can be used in environments with obstacles, but the trajectory is more difficult to control.

B. Formation Keeping Behavior

To implement the neighbor following strategies described in the previous chapter, each robot must have the knowledge of where each of its leaders should be. The knowledge of formation positions is defined for each follower robot through a configuration file and stored in an array in which each element has 4 types of information which are shown below:

```
struct Follow {
        int robotID;
        float distance;
        float angle;
```

```
        float weight;
};
```

The first item, *robotID*, is the ID of robot being followed. Item *distance* is the distance between the mass center of the robot being followed and the point where the follower robot should be located to maintain formation. Item *angle* is the relative angle of the desired formation point of the follower robot with respect to its leader robot in the leader's coordinate frame. The last item, *weight*, is used to assign different weights to each of the leader robots when multiple neighbor following is used. This way, the follower robot can have preference for some of the robots it is following. Unfortunately, the weighted average method is not used in the simulation since an appropriate method to assign weight to each leader robot is yet to be found.

In implementing the formation keeping algorithm, different strategies such as direction and speed estimation and dynamic prediction can be toggled by setting corresponding switches. For example, if direction and speed estimation is activated, the current moving direction of a leader robot is calculated by using consecutive position measurements; otherwise, inter-robot communication is assumed and the follower robot obtains the correct orientation value through communication.

In formation marching, sometimes a follower robot may find that some or all its leaders are invisible because of occlusion. If only some of its leader robots are not visible, the follower robot simply follows the remaining visible ones; otherwise the output force vector produced by the formation keeping behavior will remain the same till the last possible moment hoping its leader robots will reappear at a later time. it is worth mentioning that each robot sensor also has a maximum sensor range. If a obstacle is very large, it may happen that a robot can never sense its leader once they are separated by the obstacle. In this case formation marching fails.

C. Obstacle Avoidance Behavior

The obstacle avoidance behavior steers the robot away from nearby objects. If an obstacle sensor is activated, the obstacle avoidance behavior generates a repulsive force vector that pushes the robot away from the obstacle. Potential field method [Khatib1986, Latombe1991] is used in producing the force vector. However, as in Arkin's reactive navigation approach, the entire potential field is never computed, and only the point where the robot currently is located is calculated. If several different obstacles are detected at the same time, the total force is the vector sum of all the repulsive forces produced by individual obstacles.

D. Behavior Integration

As in Arkin's reactive navigation approach, the output of each behavior is a force vector, and the total force of the controller is the weighted sum of all the forces produced by individual behaviors. i.e.,

$$\vec{f_{output}} = \sum_{i=1}^{n} W_i \vec{f_i} \tag{4.1}$$

where $\vec{f_i}$ is the force vector produced by the ith behavior and W_i is the weight assigned to $\vec{f_i}$. Note that the weight of the formation keeping behavior is always set to 1 because the force produced by the formation-keeping behavior needs to be preserved so that there is always enough force to keep the follower robot in formation. On the other hand, the weights of other behaviors can be altered and are decided heuristics. By giving different weights to different behaviors, the motion of a robot can change significantly. For example, if the weight assigned to the avoid-obstacle behavior becomes larger, the robot will become more sensitive to the existence of obstacles.

One problem of potential field based behavior integration is that of the *local minimum* which is displayed as the lack of progress in achieving a task. For example, when a follower robot meets an obstacle, the obstacle avoidance behavior produces a

repulsive force vector while the keep formation behavior produces an attractive force vector. It may happen that the two force vectors just have opposite directions so they will cancel each other and the robot will stagnate. There are several methods to overcome this problem. One possible solution used by [Arkin1987] is to inject a force vector with random direction to the total force to move the robot out of stagnation. In our implementation, a heuristic is used to determine to which direction the robot should turn and inject a force vector perpendicular to the total force for the robot to overcome the local minima until the heading of the robot is no longer pointed to the obstacle.

4.3.3 Actuators

The actuator of the robot is defined as two separate driven wheels. The actuator receives the force vector produced by the controller and the output is the linear speeds of both left wheel and right wheel. Still, an equally distributed error within pre-specified range may exist.

```
struct Actuator {
        float leftspeed;
        float rightspeed;
        float motionerror;
};
```

4.4 User Interface

RoboSquad V2.0 runs on a Sun Sparc Workstation under SunOS and X11 graphical window system. The software is written in C. Graphic user interface is written using the Motif library from the Open Software Foundation (Two excellent introductory book on the Motif toolkit are [Heller1994] and [Brain1992]) and VOGL 3-D graphics

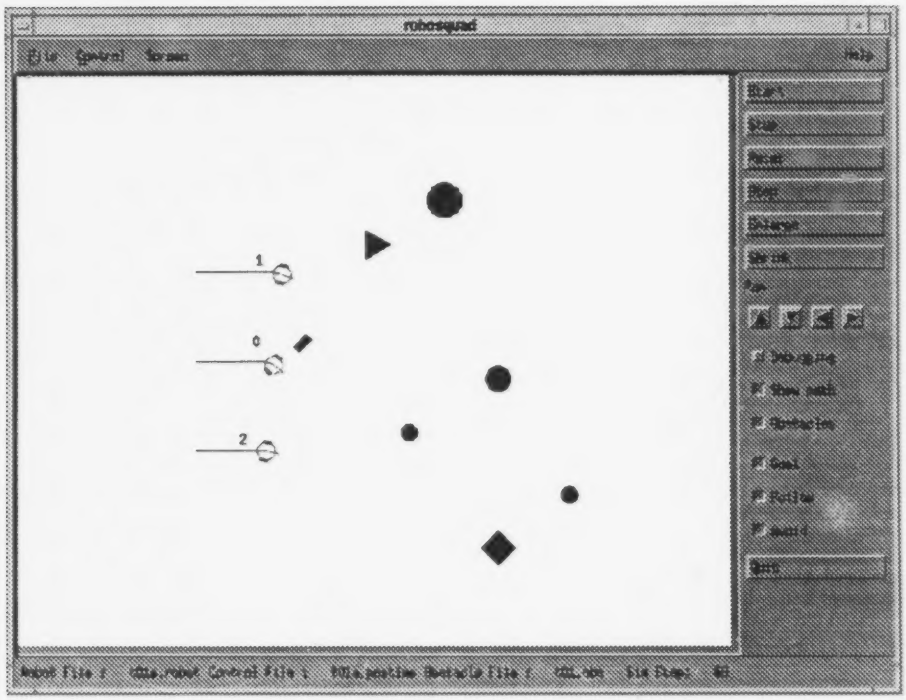


Figure 16: Graphical user interface of RoboSquad

library from University of Melbourne which is a SGI GL implementation in X-Window systems. Figure 16 shows the graphical user interface of RoboSquad V2.0.

The design principle of RoboSquad's user interface is to allow the users to test the algorithm under different settings and gather experiment data easily. Features of RoboSquad include:

- Complete command sets are available through pulldown menus while the frequently used commands such as start, stop and reset simulation, single step, enlarge and shrink display area can be directly accessed through buttons;
- Each individual behavior of the robots in the group (goal, follow, avoid) can be turned on or turned off to examine the motion of the robots under different behavior combinations;
- The *debugging* and *show path* options can be turned on to provide more infor-

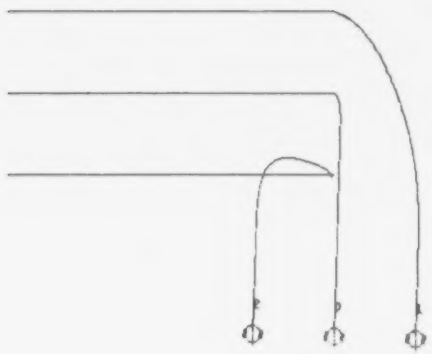


Figure 17: Simulation snapshots showing a group of 3 robots make a right turn in an obstacle-free environment

mation of the simulation process and turned off to speed up the simulation;

- The colors of different display elements (background, robots, obstacles, and robot trajectory) can be user reconfigured for both color and monochrome monitors. Also, the parameters for controlling the drawing area (enlarge/shrink scale and pan distances) can be user configured;

- On-line help is available to provide fast and convenient reference.

Figure 17 is a typical simulation snapshot showing 3 robots making a right turn in an obstacle-less environment. Robot 0 is the group leader while robot 1 and robot 2 are two follower robots trying to stay in a row with robot 0. Figure 18 is the same robot group traveling along the same path through an environment with various objects. We can see disturbance in the formation because each robot needs to avoid obstacles. However, once there are fewer obstacles near the robot group, the group resumes its assigned formation while moving along the path. A non-interactive version of RoboSquad is also developed to generate experimental data for analyzing the algorithms.

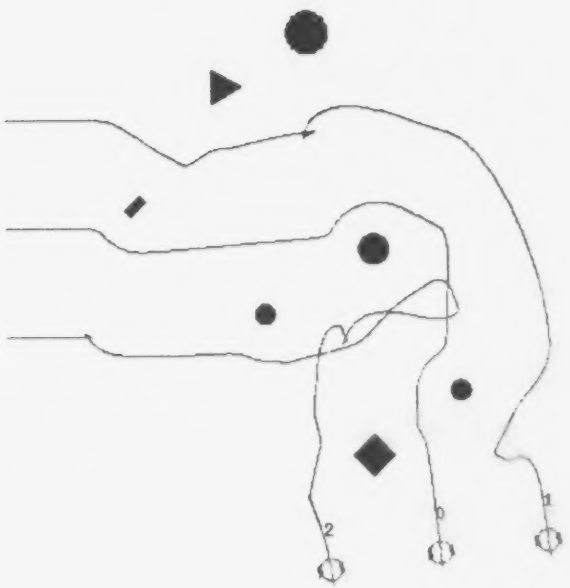


Figure 18: Simulation snapshots showing a group of 3 robots make a right turn through various obstacles

4.5 Summary

In this chapter, the simulation implementation of the neighbor following algorithm and the RoboSquad V2.0 robot population simulator are discussed. Through simulation, the performance of various neighbor following strategies under various different environments can be tested efficiently.

In the next chapter, the experimental results of the neighbor following algorithms under different test conditions are discussed to show the feasibility of the neighbor following approach.

Chapter 5

Simulation Results

5.1 Introduction

In the previous two chapters, the neighbor following approach to decentralized motion coordination of multiple robots moving in formation is discussed and the robot population simulator RoboSquad is presented. Methods to reduce static formation errors and eliminate inter-robot communications are also discussed. Multiple neighbor following method is also introduced as a means to produce more robust performance under erroneous sensor inputs. However, the methods and arguments proposed previously must be tested to see if they are valid. In this chapter, a series of simulation experiments are conducted to test both the simulator and the algorithms.

In this chapter, the concept and measurement of formation error is first introduced, then various sources of error in implementing the neighbor following approach to physical robots are analyzed, emphasis is put on sensor and sensor errors. Finally, simulation experiments are conducted to examine:

1. The effects of sensor errors on the formation performance of single and multiple neighbor following methods;

2. The effects of inter-robot communication (i.e., the true value obtained through

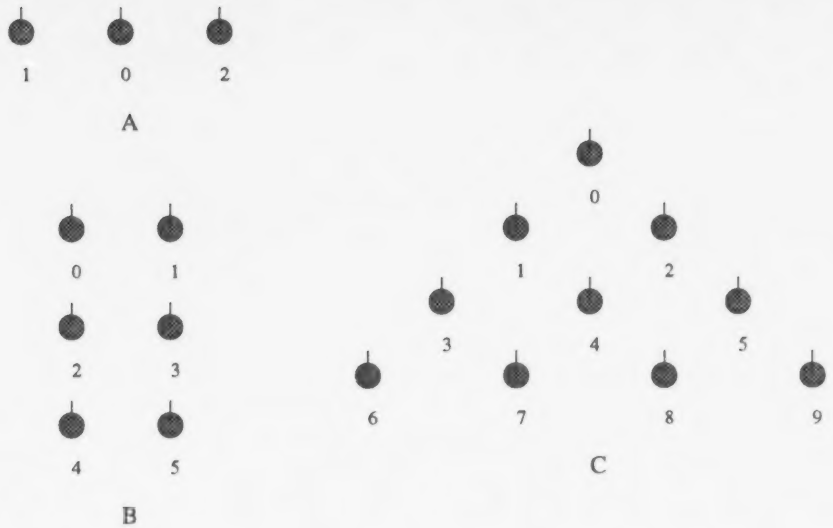


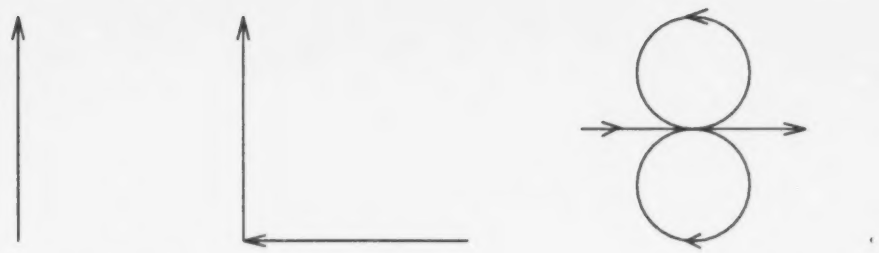
Figure 19: Three different robot formats. A: Line formation with 3 robots; B: Grid formation with 6 robots; C: Pyramid formation with 10 robots.

inter-robot communication vs. the measured value of the relative orientation and distance of a leader robot) on the formation performance of single and multiple neighbor following methods;

3. The effects of following sequence on the formation performance of single and multiple neighbor following methods.

4. The effect of obstacles on the formation performance of different formation methods.

Figure 19 illustrates three different robot group configurations, and Figure 20 illustrates three different moving trajectories of the group used in the simulation experiments. The purpose is to examine the validity of the neighbor following approach when it is applied to different robot formations and group motion trajectories.



A B C

Figure 20: Three different robot group moving trajectories. A: Straight line ("1") trajectory; B: "L" trajectory; C: "8" trajectory.

5.2 Formation Error

To test the performance of the decentralized control strategies for multiple robots moving in formation, a metric is needed to compare experiment results using different methods and under different environments. [Parker1993] defined formation error as the distance between the actual position of a follower robot and its required position with respect to the group leader, and use the cumulative formation error as the measurement of formation performance. This method is straightforward and easy to implement in simulation, but it highly depends on the motion of the group leader. For example, if there exist obstacles in the environment and the group leader invokes the obstacle-avoidance behavior to keep away from them, the robots in the group may need to move abruptly to keep the formation error minimal, and a smooth motion of the group which is normally regarded as a good formation may appear worse if the above measurement is used. A good formation error measurement should be independent of the formation of the robot group, the method of keeping the formation, the path along with the robot group is moving, and the existence of obstacles in the environment. Also, a good formation error measurement should put emphasis on the overall motion of the robot group and should be in favor of more synchronized motions of the robots in the group. Parker's method lacks the above properties, especially the latter.

In error analysis of this thesis, we treat a required or desired robot formation as a *frame*. In a frame, the frame center is the center of the robot group, which in turn is the average of the x- and y-coordinates of all the robots in the group. The direction is the intended moving direction of the robot group. This way, each robot in the group can be assigned its position in the frame, and a perfect formation means the position of each robot in the group always matches that in the frame. In real formation marching applications, the robots in the group often can not maintain their desired positions for some of the following reasons:

- When decentralized control is used, each robot decides its motion based on its sensor inputs and local communications; in this case the motions of the robots may not be highly synchronized;
- Error in sensing and actuation may cause some undesired motions of a robot;
- When the path of the group is not a straight line, the robot formation may have distortions due to the delay in the robots' reaction to the motion change of other robots;
- When multiple behaviors are activated, they may interfere with each other and may prevent a robot from keeping its desired formation position.

To measure the formation performance of a robot group moving along a certain path, we calculate the position of the group center and the moving direction of the group which is approximately the moving direction of the group center in two consecutive measurements. Then the standard frame is overlaped to the robot group by aligning the center and direction. We define, for each robot, its *instantaneous formation error* as the distance between the actual position and that in the frame, and the *average formation error* is the average of the cumulative instan taneous formation errors of all the robots in the group during the entire mission. Formally speaking, assuming in step $t(t = 1 \cdots n)$, robot $R_i(i = 1 \cdots m)$ has an *instantaneous formation error* E_{it},

then the *average formation error* E_{ave} is:

$$E_{ave} = \frac{1}{nm} \sum_{t=1}^{n} \sum_{i=1}^{m} E_{it} \tag{5.1}$$

where m is the total number of robots, and n is the total number of steps.

5.3 Error Sources in Physical Implementation

It has been shown that the single neighbor following strategy can achieve satisfactory formation performance when sensory inputs and actuation are accurate, and inter-robot communications are reliable. However, in real world, both sensors and actuators have errors, and communication between robots are not always available or reliable. Both the errors in sensing and actuation and the errors in inter-robot communication can result in the degradation or failure of the robot formation. To build a group of physical robots that can successfully move in formation in the real world, the above factors must be considered. In this thesis, we assume that communications between robots are reliable, and there is no actuation error. By using appropriate communication protocols and using feedback control in the actuation motors, those requirements are not difficult to achieve. However, sensor errors are not easy to eliminate and may greatly affect the neighbor following performance as will be discussed in the following sections.

5.3.1 Robot Sensors

Several types of robot sensors can be used to detect the relative position between a mobile robot and its leader robot. The first method is to install Global Positioning Systems (GPSs) on the robots to obtain the absolute position of each robot. Then the robots exchange through communication to obtain relative positions. GPS is usually used in outdoor applications, but the recent developments in *Pseudolites* (Pseudo Satellites) enables GPS technology to be used in indoor and urban environments

[Dowling1994]. Recent developments in differential systems and tracking techniques can achieve a real-time accuracy of millimeter range despite the precision limitation intentionally imposed by US Military. However, GPS products are expensive, especially in multi-robot applications, and the use of GPS provides extra communication overloads, thus making it unsuitable for our application.

The second method is to use optical position sensors. This type of sensors use cameras or laser scanners, to track reference targets, either actively or passively, mounted in other objects to detect the position and heading relative to the known targets. For example, [Tsumura1992] described an on-board laser scanner that measures the relative position of a leader robot from a follower robot. A laser scanner is installed on the follower robot while a cylinder covered by retro-reflecting tape is installed on the leader robot. The laser scanner rotates at a constant speed. When the laser illuminates the retro-reflecting cylinder, the photo sensor in the scanner detects the reflecting light, and the relative direction and angle of the leader robot can be calculated. If laser scanners are used in this type of sensors, the direction accuracy can reach centimeter or millimeter range in 10-meter-scale areas [Dowling1994]. However, this type of sensors are also expensive.

The third method is to use arrays of ultrasonic sensors. [McKerrow1990] described a Denning sonar ring used in the Neptune mobile robot in Carnegie-Mellon University. A ring of 24 sonar sensors are placed around the robot to measure objects in all directions. This kind of sensor arrangement is cheap to obtain and can get distance readings with accuracy up to 1% over its entire range, but the angular precision is limited due to the number of sensors allowed in the ring.

The fourth method is to place ultrasonic beacons on each robot and use rotational pairs of beacon receivers to detect the relative direction of another robot, the distance to the robot being measured is obtained using another range-finder or sonar sensor also connected to the rotational device. This method may achieve satisfactory accuracies in both direction and angle measurements. However, the ultrasonic beacons and

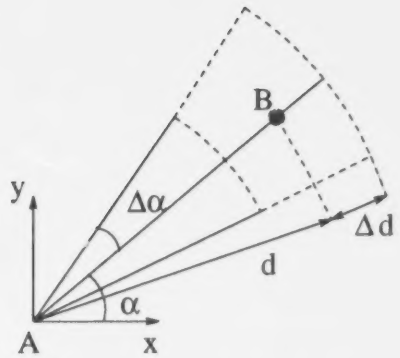


Figure 21: Sensor errors in measuring the relative distance and angle of a leader robot.

receivers need to be fine-tuned to discern different robots.

5.3.2 Sensor Errors

There are two types of sensor errors, namely, *static error* and *random error*. Static error is usually a constant value and can be eliminated by sensor calibration. Random error, however, can not be easily neutralized, and its negative effect on the performance of a robot system must be considered.

In the study of how sensory error can affect the performance of formation marching, only the errors of robot sensors are considered. Each proximity sensor reading contains errors along both the radial and angular directions. The error along the radial direction or the *distance error* Δd is the random error in measuring the distance d between the follower robot A and its specified leader robot B, and the error along the angular direction or the *angle error* $\Delta\alpha$ is the random error in measuring the angle α between the heading direction of the follower robot A and the straight line connecting the follower robot and its specified leader B. Both distance and angle errors are normally distributed random errors centered at the true values. Figure 21 illustrates how the distance and angle errors produce the resultant position error. From the figure, we can see because of Δd and $\Delta\alpha$, the exact position B of an object can not be obtained. One can only confine that the object to be inside the area

around B enclosed by dashes.

5.4 The Effects of Sensor Errors on the Performance of Robot Formation

In neighbor following, when the true values of the speed and relative heading of a leader robot are known through communication, the formation error of a follower robot is depend the errors in sensing the distance and angle error of the leader robot. If the sensing errors of each follower robot in a group are in the same level, it can be expected that the average formation error E_{ave} to be of polynomial relationship to the errors in sensing the distance and relative angles of the leader robots (not necessarily the group leader).

Experiments have been conducted on three different robot formations (see Figure 19) moving in three different trajectories (see Figure 20). They are:

1. Line formation of 3 robots moving in a "L" trajectory;
2. Grid formation of 6 robots moving in a "8" trajectory;
3. Pyramid formation of 10 robots moving in a straight line formation.

The following sequences are shown in Figure 22. Dynamic prediction is used to eliminate static position error. Since the true values of the speed and relative heading of each leader robot are known through communication, dynamic prediction will not cause formation error.

Figure 23 shows the average formation errors f different formations and trajectories with respect to different values of the variances of sensor distance errors and angle errors. The variances of sensor distance error are chosen from 0 to 0.1 square meter, while angle errors are chosen from 0 to 0.1 square radian (i.e., 0 to 328.3 square degrees). From Figure 23 we can see that the average formation error in each

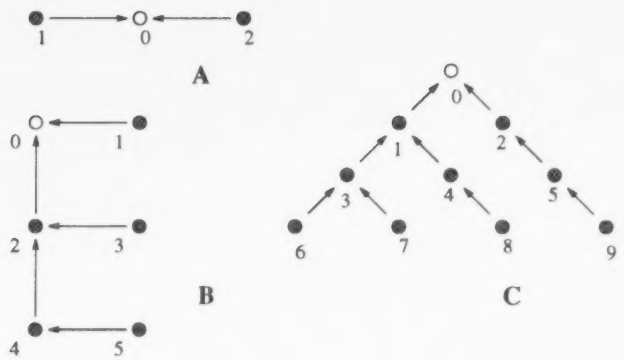


Figure 22: Three different robot groups and their neighbor following sequences. A: Three robots in a line formation; B: Six robots in a grid formation; C: Ten robots in a pyramid formation.

experiment increases near proportionally with the increase of both variance of sensor distance error and angle error. This shows that the robustness of the single neighbor following approach is good even when large sensor errors exist if moderate communication is used to provide each follower robot with necessary information about its leader's speed and relative heading. Note that the formation performance shown in Figure 23 is not comparable between each other because of the differences in group configurations and motion trajectories. Simulation experiments have also been conducted for other robot group configurations, and they all produce similar results.

5.5 The Effects of Local Communication on the Performance of Robot Formation

In chapter 3, we argued that by using leader speed and direction estimation, explicit communication between robots can be eliminated, and fully decentralized control of multiple robots moving in formation can be achieved. We also argued that using leader speed and direction puts much higher requirements on the robots' sensory systems. When the sensed position error of a leader robot becomes comparable with its location change in two consecutive position measurements, the speed and heading

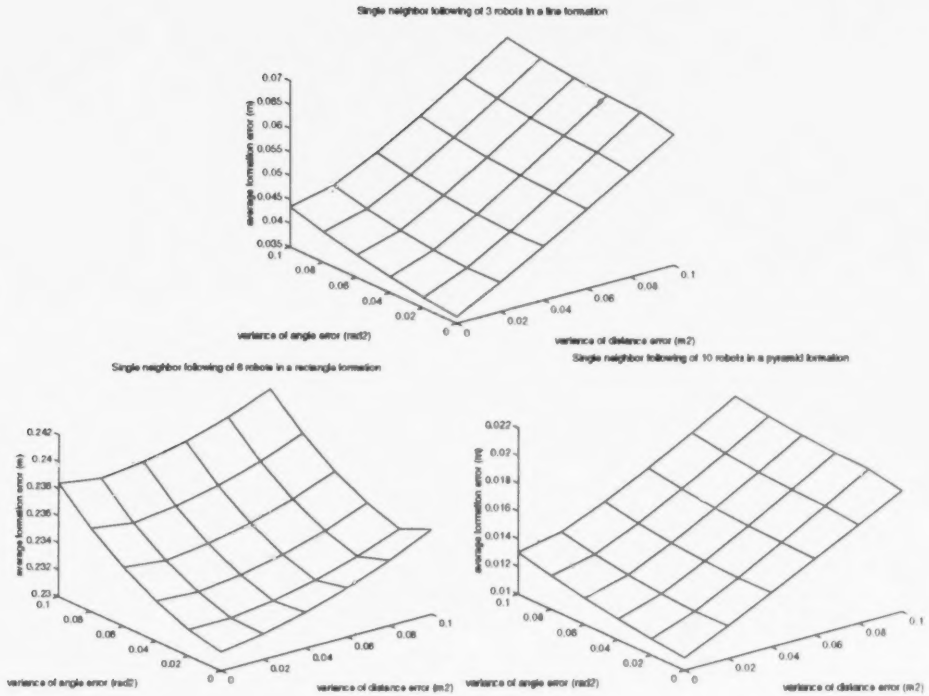

Single neighbor following of 3 robots in a line formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 10 robots in a pyramid formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)

Figure 23: Average formation errors in three different robot group moving in three different trajectories. Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation.

direction of the robot will become unpredictable, thus the formation can not be preserved. However, when the sensed leader position error is small compared with its location change in two consecutive position measurements, the formation error of the follower robot with respect to its leader robot is proportional to the sensor distance and angle errors since the speed and heading of the follower robot are calculated from the sensed distance and angle of its leader robot.

Experiments have been conducted on the same three robot group configurations and trajectories as in the previous section (see Figure 22) with the following differences:

1. Direction and speed estimations are used to get the heading direction and moving speed of each leader robot through sensing instead of using the accurate

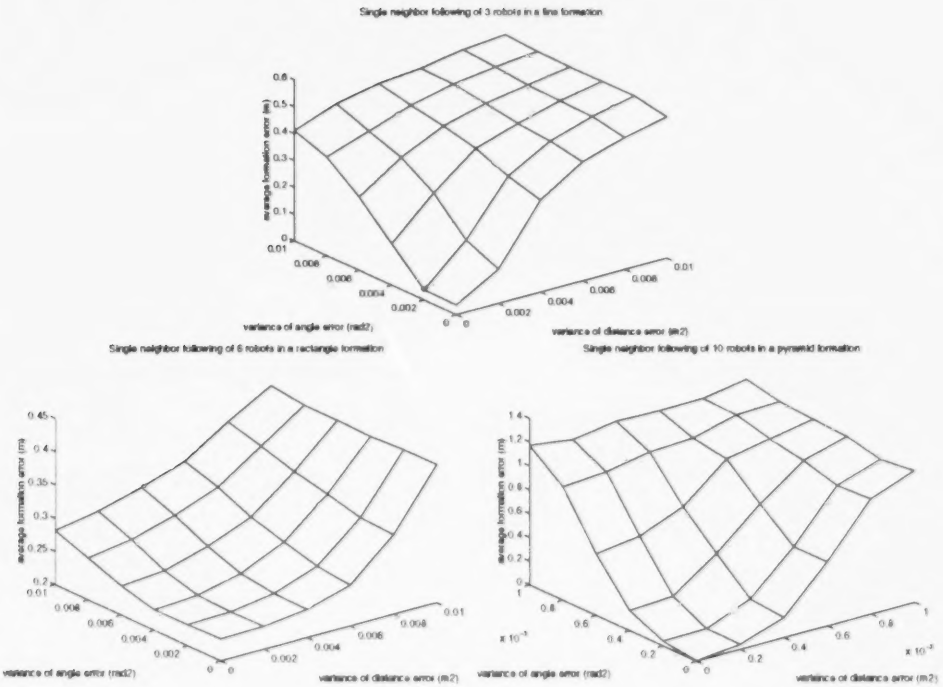

Single neighbor following of 3 robots in a line formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 10 robots in a pyramid formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)

Figure 24: Average formation errors in three different robot group moving in three different trajectories (with direction and speed estimation). Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation.

value;

2. The variances of sensor errors (distance error and angle error) are smaller than that in the previous section. For the line and grid formations, the variance of distance error is now from 0 to 0.01 square meter, while the variance of angle error is now from 0 to 0.01 square radian (0 to 32.8 square degrees). For the pyramid formation of 10 robots, the variance of distance error is now from 0 to 0.001 square meter, while the variance of angle error is now from 0 to 0.001 square radian (0 to 3.28 square degrees).

Figure 24 shows the average formation errors of the above experiments. Several observations can be drawn from the figure:

- Although sensor errors are much smaller than in the experiments using accurate leader directions and speeds, the average formation errors of all 3 cases are comparable with the formation errors of the corresponding cases shown in Figure 23;

- Because more factors can affect the formation error of the robot group when direction and speed estimation is used other than the accurate values of the heading and speed of each leader robot, the average formation error of a robot group usually grows worse than linearly with respect to sensor errors.

The above observations confirm our argument in Chapter 3 that although fully decentralized control without inter-robot communications can be achieved by using leader direction and speed estimations in neighbor following, the much higher precision requirements for the sensor system often make it impractical for many applications. Moderate inter-robot communications can greatly improve the formation performances of the groups without putting heavy communication overload on the multi-robot system. In conclusion, one must determine the trade-off between sensing and communication, depending on the application and available resources.

It is worth mentioning that in the above discussions, we treat the orientation of a leader robot obtained through communication is the true value. This is often not true. In physical implementation, each robot has to sense its heading. Depending upon the technology used, this heading may or may not be accurate. Often the accuracy of the direction is restricted by the precision of the sensing system, in this case, the average direction error E_{dir} is $E_{dir} = 0.5 * (360/n)$ where n is the number of different directions the direction sensor can discern. For example, a cheap compass may only have 3 bits precision, that is, it can only point out 8 directions. so E_{dir} is about 22.5 degrees. Since both a follower robot and its leader robot has direction measurement error, their relative orientation will also have error, and this error in relative orientation will affect the formation perfermance of a robot group.

Experiments have been conducted on the same three robot group configurations

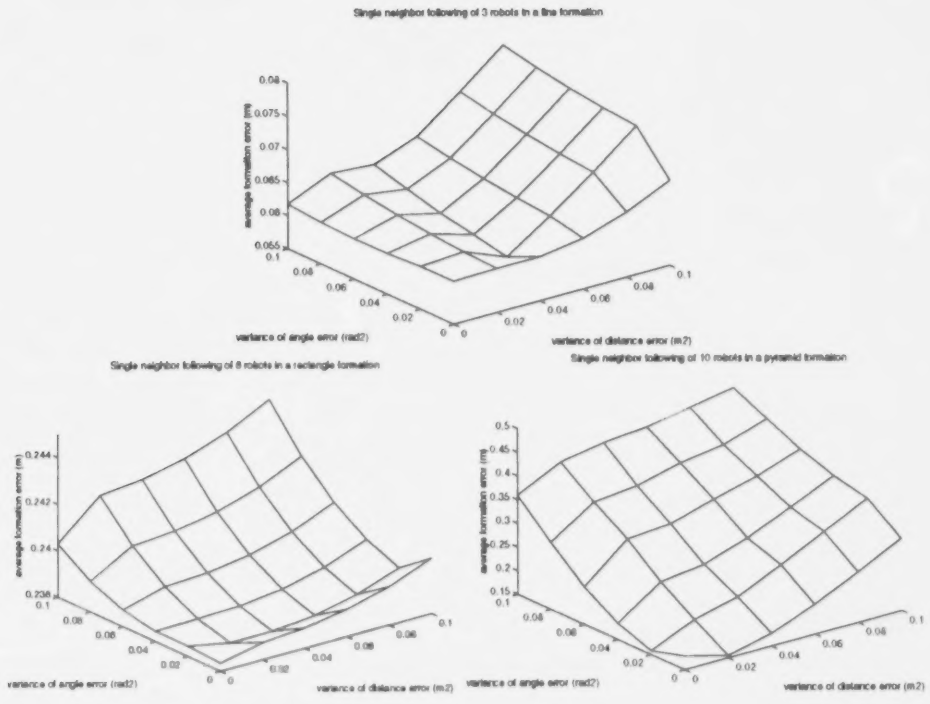

Single neighbor following of 3 robots in a line formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 10 robots in a pyramid formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)

Figure 25: Average formation errors in three different robot group moving in three different trajectories. Communication is used but error in leader directions exist. Upper: Line formation of 3 robots moving in a "L" trajectory; Bottom Left: Grid formation of 6 robots moving in a "8" trajectory; Bottom right: Pyramid formation of 10 robots moving in a straight line formation.

and trajectories as in the previous section (see Figure 22). Communication is used to send the necessary orientation information. The only difference is now there is a equally distributed random angle error in the relative orientations used in calculating following behavior for each follower robot. The error range is between -0.05 radian to 0.05 radian (about -3 degree to 3 degree). Figure 25 shows the formation perfermance of the three robot groups. When compared with Figure 23, we can see the formation performance of all three robot groups are worse when errors in relative orientation exist. However, they are still much better when compared with experiments without local communications. These results also show that local communication can greatly increase the robustness of the robot formations.

5.6 Single Neighbor Following vs. Multi-Neighbor Following

5.6.1 Formation Performances under Noisy Sensor Inputs

One objective of multiple neighbor following is to provide better formation performance of a robot group than single neighbor following under noisy sensor inputs. In Chapter 3, we argued that multiple neighbor following is superior to single neighbor following because of its "sensor fusion" or "sensor integration" effect. One interesting question is: to what extent can multiple neighbor following improve the formation performance of a robot group?

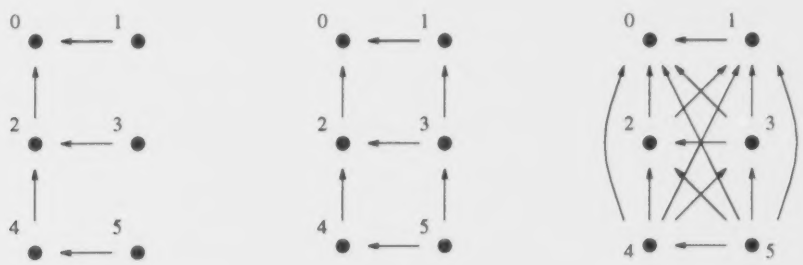


Figure 26: Different following sequences of a group of 6 robots with 1 group leader

To compare the formation performance of different neighbor following strategies with different numbers of leader robots for each follower robot. We select 3 different formation configurations which are shown in Figure 26 for the group of 6 robots in a grid formation moving along the "8" trajectory shown in Figure 22:

1. *Single neighbor following*: Robot 0 is the group leader, robot 1 and robot 2 follow robot 0, robot 3 and robot 4 follow robot 2, and robot 5 follows robot 4;

2. *Multiple neighbor following with a maximum of two leaders*: The following sequence is the same as in the single neighbor following case except that now

robot 3 also follows robot 1 and robot 5 also follows robot 4;

3. *Complete neighbor following*: Robot 0 is the group leader, each follower robot follows all its preceding robots. That is, robot 1 follows robot 0, robot 2 follows robots 0 and 1, robot 3 follows robots 0, 1, and 2, robot 4 follows robots 0 through 3, and robot 5 follows robot 0 through 4.

The average formation errors for the above three different group configurations under the same distance and angle error ranges (0 to 0.1 square meter for distance error variance, and 0 to 0.1 square radian for angle error variance) are shown in Figure 27. Experimental results show that by allowing two robots to follow two robots instead

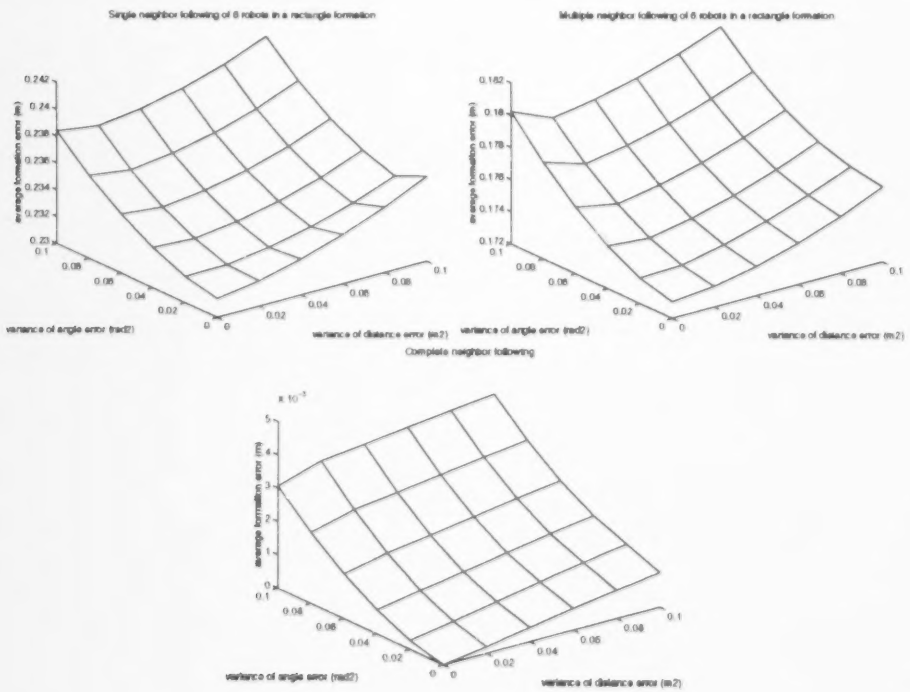

Single neighbor following of 6 robots in a rectangle formation
Multiple neighbor following of 6 robots in a rectangle formation
Complete neighbor following

Figure 27: Average formation errors in 3 different neighbor following sequences: top left: single neighbor following; top right: multiple neighbor following; bottom: complete neighbor following.

of just one robot, the formation performance improved significantly. The average formation error decreased almost an average of 25% in configuration two when compared with configuration one. However, increasing the number of leader robots for

each follower robot does not necessarily produce significant further performance improvements. For example, the average formation error decreased only an average of 4% in configuration three when compared with configuration two. The reason for this is that far-away leader robots often do not provide information useful enough to reduce measure errors. Also, occlusions occur more for far-away leader robots that make them less useful in decreasing formation error for the follower robot. When different robot formations and different trajectories are examined, the percentages may change, but the trend remains the same.

Experiments have also been conducted for robot groups using leader direction and speed estimation. When sensor errors are small enough, multiple neighbor following can also improve the formation performance of the robot group. However, as in the single neighbor following cases, when the value of sensor errors becomes comparable with the position change of the group leaders between two consecutive measurements, the formation of the robot group can not be maintained, and the average formation error becomes meaningless.

5.6.2 Formation Performances in Obstacle Filled Environment

Another objective of using multiple neighbor following is to provide more natural and coordinated group motion when obstacles exist. For a group of more than a few robots, if single neighbor following is used, it may happen that two nearby robots have vary different following sequences, thus their motion may be completely different when they or their leader robots avoid obstacles. Also, when obstacles exist, a follower robot is more likely to lost its leader because of occlusion if single neighbor following is used.

We conducted experiments on a group of 6 robots moving through an obstacle-filled area in a grid formation. Two different following sequences shown in Figure 28 are investigated, and the moving trajectory of the group is a straight line. Figure 29

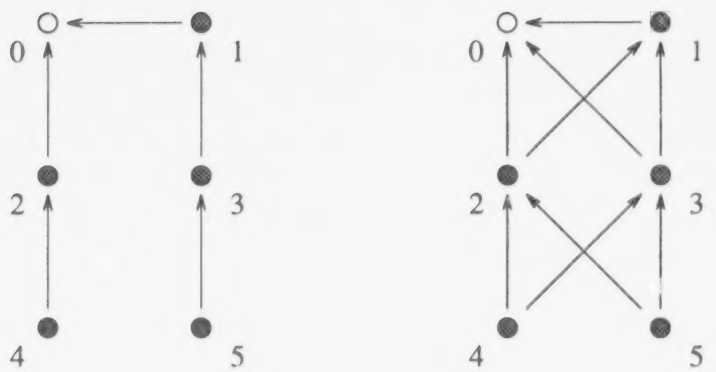


Figure 28: Two neighbor following sequences of a group of 6 robots. Left: single neighbor following; Right: multiple neighbor following.

shows the trajectories of the group of 6 robots moving through some obstacles in grid formation using the different following sequences shown in Figure 28. Although the two trajectories are similar, we can see that the motions of the robots tend to be smoother when multiple neighbor following is used (bottom figure). However, "smooth" is a rather subjective criterion to measure the formation performance of a robot group, and an objective method should be developed in future research.

5.7 The Effects of Different Following Sequences on the Performance of Robot Formation

To use the neighbor following approach on a large group of robots, the following sequence of the robots is important. As we discussed in Chapter 3, the depth of the following architecture should be as small as possible to avoid error propagation. However, the distance between a follower robot and its leaders should also be as small as possible. When designing the following sequence for a group of robots, sometimes the previous two requirements will conflict with each other. A proper balance between them must be maintained to achieve an overall optimal following sequence.

We conducted experiments on a group of 6 simulated robots moving in a grid formation along a path of shape "8". There is one group leader robot and the rest five robots are followers. In the following tests, the accurate relative orientation and velocity of each leader robot are given through communication.

Four different single neighbor following sequences as shown in Figure 30 produce different levels of formation errors, which are shown in Figure 31. It can be seen that the average formation errors of all four sequences are in linear relationship with both sensor distance error and sensor angle error. This confirms the error analysis in Chapter 3. The average formation error in the worst sequence (sequence B) is about 70% larger than that in the best sequence (sequence A) in the whole range of sensor errors tested because the former has much longer following sequence than the latter. Also, the further the distance between the leader robot and a following robot, the more sensing error the follower will get because the result position error in tangent direction is proportional to the distance under a given angular error, and occlusion may decrease the accuracy of formation. That is why although sequence D has the shortest following sequence, its formation performance is only about the same with sequence C and even worse than sequence A

5.8 Summary

In this chapter, simulation experiments are conducted on various robot groups moving along different trajectories to show the validity of the neighbor following approach under different test environments, especially when sensory errors exist. Experimental results show that with moderate inter-robot communication about each leader robot's speed and relative heading, satisfactory formation performance of the robot group can be achieved even with relative large random sensory errors for each robot. When sensing is used to determine each leader's speed and heading, the requirements for the sensor system of each robot become much higher and the formation performance of the robot group degrades much faster than that with moderate com-

munications. Experiment results also show that multi-neighbor following can produce better formation performance than single neighbor following under similar test conditions owing to the additional information provided by the additional leader robots. Finally, experiment results show that the following sequence of a robot group also plays an important role in the formation performance of the group, and a balance between various factors must be maintained to achieve optimal performance.

After showing the feasibility of the neighbor following approach of controlling multiple robots moving in formation, a brief summary of this thesis will be presented in the next chapter. Also, some future research topics will also be discussed.

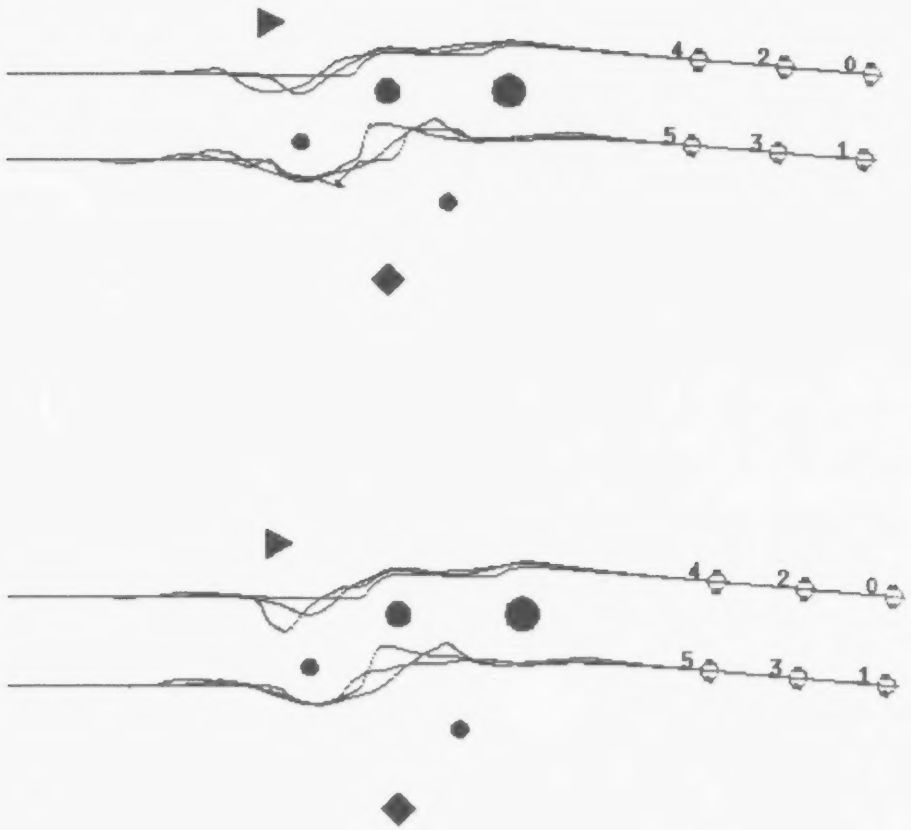


Figure 29: Simulation snapshots of a group of 6 robots moving through obstacles using two different multiple neighbor following sequences: Top: single neighbor following; Bottom: multiple (two) neighbor following.

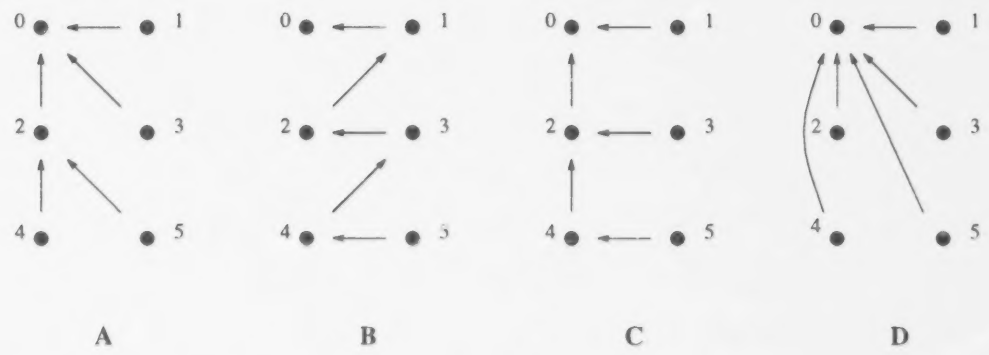


Figure 30: Different single neighbor following sequences of a group of 6 robots with 1 group leader

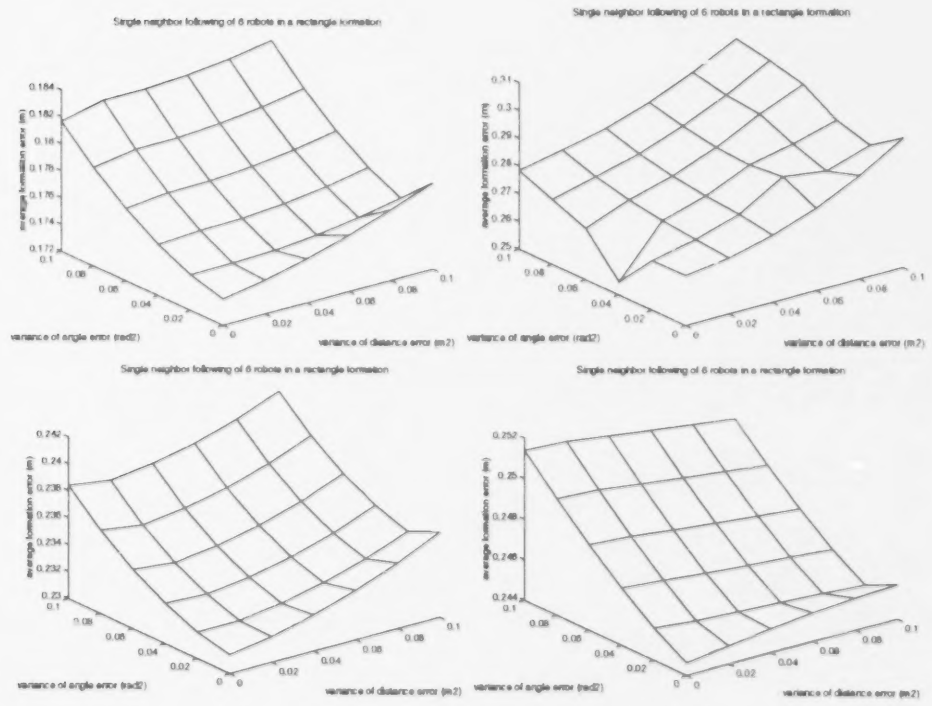

Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)
Single neighbor following of 6 robots in a rectangle formation
average formation error (m)
variance of angle error (rad2)
variance of distance error (m2)

Figure 31: Average formation errors in 4 different single neighbor following sequences.

Chapter 6

Conclusion

6.1 Summary of Thesis

In this thesis, existing approaches to control multiple mobile robots moving in formation were discussed. Then the neighbor following approach is introduced. Our neighbor following approach has several unique properties when compared with various previous approaches:

- When consecutive position measurements are used to get the heading directions and velocities of leader robots (direction estimation and dynamic prediction), our neighbor following approach becomes a fully decentralized control approach that enables a robot group moving in formation without central controller, world coordinates, or explicit inter-robot communications. Simulation experiments have shown that arbitrary formations can be achieved provided sensor inputs are accurate enough;

- Each follower robot can either follow one other robot (single neighbor following) or follow several other robots (multiple neighbor following). When sensory errors exist, the use of multiple robots as reference can effectively reduce formation errors. Also, the use multiple group leaders can reduce formation error.

Further more, when obstacles exist, multiple neighbor following can reduce the disturbance of formations;

In this thesis, the robot population simulator developed in this research, RoboSquad, is introduced. Experiments on the neighbor following strategies on groups of simulated robots are conducted for various neighbor following strategies under different sensory errors, different following sequences, different communication requirements (with or without local communications), and different environments (obstacle clustered environments vs. obstacle-free environment). Several conclusions have been reached:

- Inter-robot communications can greatly reduce the requirement for sensory systems;
- The following sequence of the robot group can greatly affect the formation performance of a robot group;
- Multiple neighbor following offers better formation performance than single neighbor following. However, the performance is sub-linear;
- Multiple neighbor following offers robust behavior in obstacle clustered environment;

In conclusion, our neighbor following is a feasible approach towards the decentralized motion coordination of multiple mobile robots moving in formation.

6.2 Future Works

There are several topics that deserve further research:

- Implementation of the neighbor following strategies in physical robot systems. Currently, a group of 10 identical small mobile robots are being constructed in

the Department of Computing Science at the University of Alberta for research in collective robotics. It is hoped that the neighbor following algorithms will be tested on these robots.

- Integration of more useful behaviors for each robot. Currently, only formation keeping behavior and avoiding behavior are implemented in each follower robots, and the control of group leader is achieved by the explicit specification through path control files. Future research should be pursued in developing more useful goal-oriented navigational behaviors for both group leaders and follower robots.

- Research in other decentralized motion coordination strategies for the formation marching problem. The group-center tracking strategy mentioned in [Balch1994a] can achieve more stable formation performance, but the center of the whole group has to be calculated for each robot. A possible approach is to devise a "local center following" approach that blends the advantages of both neighbor following and group center tracking approaches. In the local center following approach, each robot calculates the local group center of several nearby robots and tries to keep a prespecified position with the local center. However, the validity of local center following is yet to be verified.

- In the current implementation, the parameters for various behavior algorithm are given as-is, with some values even chosen arbitrarily. Future research may involve using artificial neural networks to learn various values. Also fuzzy logic may be used to produce more robust behaviors.

This thesis only serves as one attempt to develop a decentralized control strategy for task-achieving multiple mobile robots. More useful tasks and control strategies need to be investigated to further explore the application of collective robots.

Bibliography

[Arkin1992] R.C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[Arkin1987] Ronald Arkin. Motor schema–based mobile robot navigation. *The Interantional Journal of Robotics Research*, 8(4):92–112, 1987.

[Arkin1990] Ronald Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering*, pages 105–122. MIT/Elsevier, 1990.

[Asama1991] H. Asama, M. K. Habib, I. Endo, K. Ozaki, A. Matsumoto, and Y. Ishida. Functional distribution among multiple mobile robots in an autonomous and decentralized robot system. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1921 – 1926, 1991.

[Balch1994a] Tucker Balch. Motor schema-based control for multiagent robot formations. Working paper, Mobile Robot Laboratory, College of Computing, Georgia Intitute of Technology, June 1994.

[Balch1994b] Tucker Balch and Ronald Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1:1–25, 1994.

[Beer1990] R. D. Beer, H. J. Chiel, and L. S. Sterling. A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6(1):169–186, 1990.

[Beni1992] G. Beni and S. Hackwood. Cyclic swarms. In 1, editor, *JAPAN/USA Symposium on Flexible Automation*, pages 655–663, 1992.

[Brain1992] Marshall Brain. *MOTIF Programming: The Essentials .... and More*. Digital Press, 1992.

[Brock1992] David L. Brock, David J. Montana, and Andrew Z. Ceranowicz. Coordination and control of multiple autonomous vehicles. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, volume 3, pages 2725–2730, Nice, France, May 1992.

[Brooks1990] R. A. Brooks. Elephants don't play chess. In P. Maes, editor, *Designing Autonomous Agents*, pages 3–15. MIT/Elsevier, 1990.

[Brooks1986] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, April 1986.

[Brooks1991b] R.A. Brooks. Intelligence without reason. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, volume 1, pages 569–595. Morgan Kaufmann, 1991.

[Brooks1991a] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–160, 1991.

[Chen1994b] Q. Chen and J. Y. S. Luh. Distributed motion coordination of multiple robots. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 3, pages 1493–1499, Munich, Germany, September 1994.

[Chen1994a] Q. Chen and J.Y.S. Luh. Coordination and control of a group of small mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2315–2320, May 1994.

[Connell1990] J. H. Connell. *Minimalist Mobile Robotics: A colony-style Architecture for an Artificial Creature.* Academic Press, Inc., San Diego, CA 92010, 1990.

[Dario1991] P. Dario, F. Ribechini, V. Genovese, and G. Sandini. Instinctive behaviors and personalities in society of cellular robots. In *Proceedings of the 1991 International Conference on Robotics and Automation*, pages 1927–1932, Sacramento, CA, April 1991.

[Dudek1993] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for swarm robots. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, July 1993.

[Durrant-Whyte1987] Hugh F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems.* Kluwer Adademic Publishers, 1987.

[Franks1989] M. R. Franks. Army ants: A collective intelligence. *American Scientist*, 77(2):139–145, March 1989.

[Fukuda1989] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss. Structure decision for self organising robots based on cell structures

- cebot. In *Porceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 695–700, 1989.

[Gat1992] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the 1992 AAAI Conference*, pages 809–815, 1992.

[Genovese1992] V. Genovese, L. Odetti, R. Magni, and P. Dario. Self organizing behavior and swarm intelligence in a cellular robotic system. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 243–248, 1992.

[Gould1974] L. L. Gould and F. Heppner. The vee formation of canada geese. *Auk*, 91:494–506, July 1974.

[Hager1990] Gregory D. Hager. *Task-Directed Sensor Fusion and Planning: A Computational Approach.* Kluwer Academic Publishers, 1990.

[Heller1994] Dan Heller and Paula M. Ferguson. *Motif Programming Manual.* O'Reilly & Associates, Inc., 2 edition, 1994.

[Heppner1985] F. H. Heppner, J. L. Convissar, D. E. Jr. Moonan, and Anderson J. G. T. Visual angle and formation flight in canada geese (branra canadensis). *Auk*, 102:195–198, January 1985.

[Heppner1974] F.H. Heppner. Avian flight formation. *Bird Banding*, 45(2):160–169, 1974.

[Ishida1991] Y. Ishida, H. Asama, I. Endo, K. Ozaki, and A. Matsumoto. Communication and cooperation in an autonomous and decentralized robot system. In *International Symposium on Dis-*

tributed Intelligent Systems (IFAC DIS'91), pages 189–194, 1991.

[Dowling1994] etc. Kevin Dowling. Usenet newsgroup comp.robotics faq, 1994.

[Khatib1986] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Jouranl of Robotics Research*, 5(1):90–98, 1986.

[Kube1992a] C.R. Kube and H. Zhang. Collective robotic intelligence. In *Proceedings of Second International Conference on Simulation of Adaptive Behavior*, pages 460–468, December 1992.

[Kube1994] C.R. Kube and H. Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–219, 1994.

[Kube1992b] R. Kube. Collective robotic intelligence: A control theory for robot populations. Master's thesis, University of Alberta, Edmonton, Alberta, Canada, April 1992.

[Latombe1991] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[Luo1989] R. C. Luo and M. G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–931, September 1989.

[MacKenzie1993] D. C. MacKenzie and R. C. Arkin. Formal specification for behavior-based mobile robots. In *SPIE Conference on Mobile Robots VIII*, pages 94–104, September 1993.

[Mataric1992] Maja J. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proceedings of the 1992 IEEE Inter-*

national Conference on Robotics and Automation, volume 1, pages 830–835, Nice, France, May 1992.

[McKerrow1990] P. J. McKerrow. *Introduction to Robotics.* Addison-Wesley Publishing Company, 1990.

[Noreils1990] F. Noreils. Integrating multirobot coordination in a mobile-robot control system. In *Proceedings of the 1990 IEEE International Workshop on Intelligent Robots and Systems(IROS)*, volume 2, pages 43–49, 1990.

[Noreils1992] F.R. Noreils. Multi-robot coordination for battlefield strategies. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1777–1784, Releigh, NC, July 1992.

[Parker1993] L. E. Parker. Designing control laws for cooperative agent teams. In *Proceedings of IEEE International Conferencence on Robotics and Automation*, volume 3, pages 582–587, 1993.

[Payton1990] D. W. Payton. Internalized plans: A representation for action resources. *Robotics and Automation Systems*, 6(1):89–104, 1990.

[Pomeroy1992] H. Pomeroy and F. Heppner. Structure of turning in airborne rock dove (columba livia) flocks. *Auk*, 109(2):256–267, 1992.

[Stilwell1994] D.J. Stilwell and J.S. Bay. Optimal control for cooperating mobile robots bearing a common load. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 58–63, San Diego, CA, May 1994.

[Sugihara1990] K. Sugihara and I. Suzuki. Distributed motion coordination of multiple mobile robots. In *Proceedings of the 5th IEEE In-*

ternational Symposium on Intelligent Control, volume 1, pages 138–143, September 1990.

[Tsumura1992] Toshihiro Tsumura, Hiroshi Okubo, and Nobuo Komatsu. Directly follow-up and/or traced control system of multiple ground vehicles. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1769–1776, July 1992.

[Ueyama1992] T. Ueyama, T. Fukuda, and F. Arai. Configuration of communication structure for distrubuted intelligent robotic system. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, volume 1, pages 807–812, May 1992.

[Unsal1993] Cem Unsal. Self-organization in large populations of mobile robots. Master's thesis, Virginia Polytechnic Institute and State University, May 1993.

[Wang1989] J. Wang and G. Beni. Pattern generation in cellular robotic system. In *Proceedings of the 4th IEEE Symposium on Intelligent Control*, pages 63–69, 1989.

[Wang1991] P.K.C. Wang. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[Ward1979] P. Ward. Formation flying as advertisment behavior. *Animal Behavior*, 27, 1979.

[Williams1976] T. C. Williams, T. J. Klonowski, and P. Berkeley. Angle of canada goose v flight formation measured by radar. *Auk*, 93:554–559, July 1976.

[Yin1992] Q. Yin and Y. F. Zheng. Performance analysis of token bus lan in coordinating multiple robots. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, volume 1, pages 455–460, Nice, France, May 1992.

[Yoshida1994] E. Yoshida, T. Arai, J. Ota, and T. Miki. Effect of grouping in local communication system of multiple mobile robots. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 2, pages 808–815, Munich, Germany, September 1994.